Lattice Diamond 3.12 Help



November 25, 2020

Copyright

Copyright © 2020 Lattice Semiconductor Corporation. All rights reserved. This document may not, in whole or part, be reproduced, modified, distributed, or publicly displayed without prior written consent from Lattice Semiconductor Corporation (“Lattice”).

Trademarks

All Lattice trademarks are as listed at www.latticesemi.com/legal. Synopsys and Synplify Pro are trademarks of Synopsys, Inc. Aldec and Active-HDL are trademarks of Aldec, Inc. Modelsim and Questa are trademarks or registered trademarks of Siemens Industry Software Inc. or its subsidiaries in the United States or other countries. All other trademarks are the property of their respective owners.

Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS “AS IS” WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF LATTICE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Lattice may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. Lattice makes no commitment to update this documentation. Lattice reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. Lattice recommends its customers obtain the latest version of the relevant information to establish that the information being relied upon is current and before ordering any products.

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of code.
`.` `.` `.`	Omitted lines in code and report examples.
`[ ]`	Optional items in syntax descriptions. In bus specifications, the brackets are required.
`( )`	Grouped items in syntax descriptions.
`{ }`	Repeatable items in syntax descriptions.
`|`	A choice between items in syntax descriptions.

Contents

Chapter 1 Help for Lattice Diamond 12
Getting Help **13**
Updating Diamond from the Web **17**
Copyright, Trademarks, and Disclaimers **20**

Chapter 2 Managing Projects 22
Running Lattice Diamond **23**
Managing Diamond Layout **24**
Creating a New Project **24**
Modifying a Project **25**
Importing ispLEVER Projects **26**
Targeting a Device **28**
Viewing Project Properties **29**
Saving Project Files **29**
Tips for Defining Projects **32**
Tips for Saving and Naming Projects **32**
Reserved File Names **32**
Managing Project Sources **34**
Working with Implementations **46**
Using Strategies **48**
Analyzing a Design **54**
Running Processes **70**
Setting Security Options **76**
Setting Options for Synthesis and Simulation **87**
Working with Run Manager **94**
Finding Results **100**
Viewing Logs and Reports **101**

Chapter 3 Designing with Lattice Diamond Platform Designer 108
About Platform Designer **112**
Platform Designer Flow **114**
Launching Platform Designer **116**
Working with Platform Designer Project (.ptm) Files **118**
Working with Platform Designer Editors **120**
Setting Global Options **123**
Configuring Voltage Sense (VMON) and Control (TRIM) **126**
Configuring Current Sense (IMON) **133**
Configuring Temperature Sense (TMON) **134**

Configuring Fan Control **135**
Setting Fault Logging Options **136**
Configuring Hot Swap **138**
Configuring PMBus Adapter **142**
Implementing the Platform Management Algorithm **143**
Building the Design **157**
Simulating the Design **160**
Working with Power Calculator **163**

Chapter 4 Entering the Design 165
HDL Design Entry **167**
Designing with Modules **190**
Creating Clarity Designer Modules **211**
Designing with Lattice Propel **239**
Designing with LatticeMico Platforms **241**
Schematic Design Entry **243**

Chapter 5 Simulating the Design 310
Simulation in Diamond **310**
Timing Simulation **314**
Third-Party Simulators **315**
Using the Global Set/Reset (GSR) Signal **341**
Creating Waveform Test Stimulus **342**

Chapter 6 Applying Design Constraints 360
Multiple Entry Constraint Flow **360**
Using SDC Constraints **364**
Using Preferences **375**
Using Diamond's Preference Views **406**
Setting Preferences **459**
Saving Preferences **546**
Discarding In-Memory Preference Changes **549**
Checking Design Rules **549**
Analyzing SSO **559**
Exporting/Importing Pin Files and Preference Sheets **564**
Adding FPGA Attributes to HDL **569**
Achieving Timing Closure **571**

Chapter 7 Implementing the Design 572
Validating an I/O Plan **573**
Synthesizing the Design **578**
Translating the Design Database **588**
Mapping **590**
Place and Route **602**
Bit Generation **639**

PROM Generation **648**

Chapter 8 Using Incremental Design Flow 651
Running the Incremental Design Flow **655**
Using Partition Manager **664**
Floorplan View and Physical View Partition Support **674**
Recommended Strategies for Incremental Design **677**
Troubleshooting Error and Warning Messages **685**

Chapter 9 Analyzing Static Timing 694
Static Timing Analysis Tools **694**
Strategies for Timing Analysis **695**
Running TRACE **696**
Running I/O Timing Analysis **722**
Using Timing Analysis View **732**

Chapter 10 Analyzing Power Consumption 756
Starting Power Calculator from Diamond **757**
Starting Power Calculator as a Stand-Alone Tool **757**
Running Power Calculator from the Tcl Console **759**
Power Analysis Design Flow **759**
Inputs **760**
Outputs **760**
Static and Dynamic Power Consumption **761**
Activity Factor Calculation **761**
Power Calculator Window Features **761**
Working with Power Calculator Files **771**
Entering Data **778**
Reverting to Calculation Mode **782**
Changing the Global Default Activity Factor **783**
Importing a Value Change Dump (.vcd) File **783**
Changing the Global Default Frequency Setting **784**
Estimating Resource Usage **785**
Estimating Routing Resource Usage **786**
Controlling Operating Temperature **786**
Controlling Power Options for Low-Power Devices **790**
Comparing Power Consumption Among Multiple Implementations **794**
Viewing and Printing Results **795**

Chapter 11 Analyzing Signal Integrity 798
Lattice Semiconductor IBIS Models **798**
HSPICE Models for FPGA Devices **801**

Chapter 12 Programming the FPGA 802
File Formats **803**

Serial Peripheral Interface (SPI) Flash Support **804**
Programming Lattice Mature Devices with Stand-Alone Diamond Programmer **816**
Programming Lattice Mature Devices with Stand-Alone Diamond Programmer **818**
Embedded Software Device Support **820**
Using Diamond Programmer **824**
Programmer Options **856**
Deploying the Design with the Deployment Tool **950**
Debugging SVF, STAPL, and VME Files **1008**
Download Debugger Options **1019**
Using the Model 300 Programmer **1022**
Model 300 Options **1031**
Using Programming File Utility **1033**
Programming File Utility Options **1038**

Chapter 13 Testing and Debugging On-Chip 1040
About Reveal Logic Analysis **1041**
Debugging Software **1043**
Creating Reveal Modules **1044**
Performing Logic Analysis **1073**

Chapter 14 Applying Engineering Change Orders 1102
Editing sysIO and PLL Preferences in ECO Editor **1102**
Setting Memory Initialization Values in ECO Editor **1104**
Modifying SerDes Settings **1109**
Adding Signal Probes **1109**
Tracking NCD Changes **1111**
Exporting sysIO Preferences **1111**
Running Design Rule Check **1111**

Chapter 15 Strategy Reference Guide 1112
Synplify Pro Options **1117**
Precision Options **1124**
LSE Options **1129**
Translate Design Options **1138**
Map Design Options **1140**
Map Trace Options **1142**
Place & Route Design Options **1144**
Place & Route Trace Options **1151**
IO Timing Analysis Options **1153**
Timing Simulation Options **1153**
Bitstream Options **1155**

Chapter 16 Hardware How-To 1162
How to Design with FPGA Memories **1162**
How to Use Dedicated DDR Memory Support **1165**
How to Design with PCS/SERDES **1166**
How to Design with sysCLOCK PLLs and DLLs **1168**
How To Design with sysIO Buffers **1171**
How to Design with sysDSP **1172**
How to Use the Global Set/Reset (GSR) Signal **1174**
How to Use the Power Up Set/Reset (PUR) Global Signal **1189**
How to Use the Tristate Interface (TSALL) Global Signal **1190**
How to Use the Internal Oscillator **1192**

Chapter 17 Constraints Reference Guide 1194
Preferences **1195**
HDL Attributes **1282**
Lattice Synthesis Engine (LSE) Constraints **1339**

Chapter 18 Lattice Module Reference Guide 1400
Finding Modules in This Guide **1400**
1D_FILTER **1403**
Adder **1404**
Adder_Subtractor **1407**
ADDER_TREE **1410**
BARREL_SHIFTER **1410**
Comparator **1411**
Complex_Multiplier **1412**
Convert **1414**
Counter **1415**
DDR **1417**
DDR_GENERIC **1419**
DDR_MEM **1428**
Digital CDR **1435**
Distributed_DPRAM **1436**
Distributed_ROM **1439**
Distributed_SPRAM **1442**
DLL **1445**
DQS **1447**
Dynamic Bank Controller **1447**
EFB **1448**
EXTREF **1457**
FFT_Butterfly **1457**
FIFO **1458**
FIFO_DC **1462**
GDDR_7:1 **1475**

I2C **1476**
LFSR **1478**
MAC **1479**
MIPI_DPHY **1491**
MMAC **1492**
MULT **1492**
Mult_Add_Sub **1500**
Mult_Add_Sub_Sum **1504**
MULTADDSUB **1507**
MULTADDSUBSUM **1526**
Multiplier **1544**
Multiply_Accumulate **1552**
ORCAstra **1555**
PCS **1556**
PLL **1573**
PMU **1600**
Power Controller **1600**
Power Guard **1601**
RAM_Based_Shift_Register **1602**
RAM_DP **1605**
RAM_DP_TRUE **1614**
RAM_DQ **1625**
ROM **1634**
SDR **1638**
Sin-Cos_Table **1643**
SLICE **1644**
Subtractor **1656**
System_Bus **1659**
Tag Memory **1663**
WIDE_MUX **1663**

Chapter 19 FPGA Libraries Reference Guide 1665
Naming Conventions **1667**
Memory Primitives Overview **1668**
RAM_DP (Dual Port RAM) **1669**
RAM_DP_BE (Dual Port RAM with Byte Enable) **1671**
RAM_DP_TRUE (True Dual Port RAM) **1672**
RAM_DP_TRUE_BE (True Dual Port RAM with Byte Enable) **1674**
RAM_DQ (Single Port RAM) **1675**
RAM_DQ_BE (Single Port RAM with Byte Enable) **1676**
ROM (Read Only Memory) **1678**
Distributed_DPRAM (Distributed Dual Port RAM) **1679**
Distributed_ROM (Distributed Read Only Memory) **1680**

Distributed_SPRAM (Distributed Single Port RAM) **1680**
FIFO (First In First Out Single Clock) **1681**
FIFO_DC (First In First Out Dual Clock) **1682**
Shift Registers (Distributed RAM Shift Register) **1684**
Primitive Library - ECP5 **1686**
Primitive Library - LatticeECP/EC and LatticeXP **1695**
Primitive Library - LatticeECP2/M **1707**
Primitive Library - LatticeECP3 **1717**
Primitive Library - LatticeSC/M **1727**
Primitive Library - LatticeXP2 **1739**
Primitive Library - LFMNX **1749**
Primitive Library - LIFMD **1759**
Primitive Library - LIFMDF **1766**
Primitive Library - LIFMDF **1773**
Primitive Library - MachXO and Platform Manager **1780**
Primitive Library - MachXO2 and Platform Manager 2 **1788**
Primitive Library - MachXO3D **1799**
Primitive Library - MachXO3L **1809**
Alphanumeric Primitives List **1819**
Primitive-Specific HDL Attributes **2445**

Chapter 20 Command Line Reference Guide 2464
Command Line Basics **2466**
Command Line Tool Usage **2477**

Chapter 21 Tcl Command Reference Guide 2586
Running the Tcl Console **2587**
Launching the EPIC Tcl Console **2588**
Accessing Command Help in the Tcl Console **2589**
Creating and Running Custom Tcl Scripts **2590**
Running Tcl Scripts When Launching Diamond **2592**
Diamond Tool Tcl Command Syntax **2593**

Chapter 22 Glossary 2636
A **2636**
B **2639**
C **2641**
D **2647**
E **2650**
F **2653**
G **2656**
H **2657**
I **2658**
J **2662**

L **2663**
M **2667**
N **2671**
O **2673**
P **2674**
Q **2683**
R **2683**
S **2685**
T **2693**
U **2697**
V **2698**
W **2699**
X **2700**
Y **2700**

Chapter 23 Design Tool Reference 2701

Chapter 1

Help for Lattice Diamond

Lattice Diamond™ is the complete design environment for Lattice Semiconductor FPGAs. The software includes a comprehensive set of tools for all design tasks, including project management, design entry, simulation, synthesis, place and route, in-system logic analysis, and more.

To start learning about Diamond, see Lattice Diamond 3.12 User Guide.

To quickly get some hands-on experience, try the Lattice Diamond 3.12 Tutorial.

To help you experiment further, example projects are available. These examples are designed to illustrate different aspects of designing with Diamond. To see the examples, choose **File > Open > Design Example**. Then open an example folder and open the example's .ldf file. (An .ldf file defines a design project for Diamond.)

The rest of the Help system provides complete instructions for the different stages of the Diamond design flow and extensive reference material.

See Also

- "Getting Help" on page 13
- Lattice Diamond Software 3.12 Release Notes

User Guides

- Lattice Diamond 3.12 User Guide
- Lattice Synthesis Engine for Diamond User Guide
- Lattice Diamond 3.12 Platform Designer User Guide
- Clarity Designer User Guide
- Lattice Diamond 3.12 Programming Tools User Guide
- Reveal User Guide
- Reveal Troubleshooting Guide
- Lattice Diamond 3.12 FPGA Libraries Reference Manual (PDF version)

Other Tool Guides

-

- ModelSim User's Manual
- ModelSim GUI Reference Manual
- ModelSim Command Reference Manual
- ModelSim Tutorial
- Synplify and Synplify Pro for Lattice User Guide
- Synplify and Synplify Pro for Lattice Reference Manual
- Synplify and Synplify Pro for Lattice Language Support Reference Manual
- Tcl/Tk Documentation

Tutorials

- Lattice Diamond 3.12 Tutorial
- Lattice Synthesis Engine Tutorial

FPGA Design Guide

- Design Planning
- HDL Coding Guidelines
- Timing Closure

Lattice on the Web

- Lattice Semiconductor
- Answer Database
- Lattice Solutions
- Intellectual Property

Getting Help

For almost all questions, the place to start is this Help. It describes the FPGA design flow using Diamond, the libraries of logic design elements, and the details of the Diamond design tools. The Help also provides easy access to many other information sources.

To make the most effective use of the Help, please review this section.

Opening the Help

The Help can be opened in several ways:

- In Windows, choose **Start > Programs > Lattice Diamond > Accessories > Diamond Help**.
- In the main window, choose **Help > Lattice Diamond Help**.
- To go directly to the Help for the tool that you're using do one of the following:

- If the tool is attached to the main window, choose **Help >** <*Tool Name*> **Help**.
- If the tool is detached, choose **Help >** **Help**.

- To go directly to the Help for the view of the tool that you're using, press **F1**.
- For a description of how to use a dialog box, click **Help** (if available).
- For a description of a feature in a dialog box, right-click on the feature. If a box appears saying "What's This?," click it. Or click on the feature and then press **Shift-F1**. In Windows, you can also click the question mark button in the title bar and then hover over the feature. If the cursor turns into an arrow with a question mark, click on the feature.

  Note: Not all dialog box features have pop-up descriptions.
- To learn more about a property in the Strategies dialog box, select the property and press **F1**.
- To get more information related to a message, right-click the message. If a menu appears with a Help command available, choose it.

JavaScript must be enabled in your default browser. If you are have trouble opening the Help, see "Troubleshooting the Help" on page 15.

Using the Help

The Help has several features to help you find information.

Contents The contents organizes the information in the Help. Look here to see what subjects are covered or to begin in-depth study of a subject.

Click the (Menu) button to hide or show the contents pane.

Search Just start typing in the toolbar. The Search page opens after four characters with a list of topics. Keep typing to shorten the list.

Search results are ordered by the number and types of hits found. The first few topics are most likely to have substantial information on the search terms. Topics at the end of the list may have just a casual mention of the terms.

Some tips:

- To find plurals and different verb tenses, just type the beginning of the word. For example, "search" finds search, searches, searching, and searched.
- Capitalization does not matter.
- Enter phrases between a pair of double-quote marks. For example: "block module". Sometimes you may see search hits that do not exactly match your phrase. This is because the search ignores certain common words like "for" and "the." So the example would also find "block the module."

- Click **All** next to the Search field and select **User Guides** or **Reference Guides** to limit the scope of the search.

Search Navigation Tip

Internet Explorer's forward and back buttons may not always work the way you expect with the Search page. To go back to the Search page, type in the toolbar. Just add a letter or space, or backspace. If you're on the Search page and want to go back to the topic you were looking at, click the next to the Search field.

Glossary The "Glossary" on page 2636 has definitions of hundreds of technical terms and abbreviations. The Glossary is in the contents under "Reference Guides."

Design Tool Reference "Design Tool Reference" on page 2701 is an index of all the tools in the Diamond tool set. This can be an easy way to find information about a tool. Design Tool Reference is in the contents under "Reference Guides."

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of code.
. . .	Omitted lines in code and report examples.
`[ ]`	Optional items in syntax descriptions. In bus specifications, the brackets are required.
`( )`	Grouped items in syntax descriptions.
`{ }`	Repeatable items in syntax descriptions.
`|`	A choice between items in syntax descriptions.

Troubleshooting the Help

If you are have trouble opening the Help, check for the following situations:

Active Content or Scripts Are Blocked Opening the online Help may be interrupted by one of the following messages on the Internet Explorer Information Bar:

- "To help protect your security, Internet Explorer has restricted this file from showing active content that could access your computer. Click here for options..."
- "To help protect your security, Internet Explorer has restricted this file from running scripts or ActiveX controls that could access your computer. Click here for options..."

To see the Help, click on the Information Bar and choose **Allow Blocked Content**. A dialog box with an expanded warning opens. Click **Yes**.

To avoid these warnings, either use a different browser or turn off the warning for active content in Internet Explorer.

Note

Doing either of these means that when you open any Web page that is resident on your computer—not just Diamond Help—the page will automatically run any active content that it has. While active content is common and can be very useful, malicious content can damage your files. Be sure you trust the software on your computer.

To turn off the warning:

1. In Internet Explorer, choose **Tools > Internet Options**.
2. Click the **Advanced** tab.
3. Under Security, select **Allow active content to run in files on My Computer**.
4. Click **OK**.

Back and Forward Buttons Do Not Work with the Search Page Internet Explorer's back and forward buttons may always not work the way you expect with the Search page. Instead of returning to the Search page, you go to the previous or next topic. Instead of returning from the Search page to the previous or next topic, you stay on the Search page. The Search page is not in the stack of visited pages.

To go back to the Search page, type in the toolbar. Just add a letter or space, or backspace. If you're on the Search page and want to go back to the topic you were looking at, click the  next to the Search field.

Contacting Technical Support

FAQs The first place to look. The Answer Database provides solutions to questions that many of our customers have already asked. Lattice Applications Engineers are continuously adding to the Database.

Technical Support Request Submit a Technical Support Request through www.latticesemi.com/techsupport.

For Local Support Contact your nearest Lattice Sales Office.

Updating Diamond from the Web

You can update Diamond from the Lattice Web site with UPDATE. The UPDATE program can be set to automatically check for service patches and alert you when one is available. You can also use UPDATE to manually check for service patches. When you find a patch you can download it and install it at your convenience. Or, in Windows, you can have the patch automatically installed after downloading it.

Start by setting the UPDATE options. See "Setting Options for Updates" on page 17.

See Also

- "Getting a Service Patch" on page 18
- "Installing a Downloaded Service Patch" on page 19

Setting Options for Updates

To get service patches, you must specify the Internet settings and if you want the software to check for service patches. These options can be set in the Diamond main window or in the UPDATE program. The following instructions are for the Diamond main window. In UPDATE, start by choosing **Settings > Update Settings**.

To set update options:

1. In Diamond's main window, choose **Tools > Options**.

   The Options dialog box opens.

2. In the hierarchy tree of the Options dialog box, choose **Environment/ Startup**.

3. If you want the software to automatically check for patches, select **Automatic check for software update when Lattice Diamond launches**, and then select the desired interval.

   Clear this option to disable automatic checking. If you disable this option, you should occasionally run a manual check as described in "Getting a Service Patch" on page 18.

4. In the hierarchy tree of the Options dialog box, choose **Environment/ Network Settings**.

   You need to set up an Internet connection to download patches regardless of whether you selected automatic checking.

5. If you use a proxy server to reach the Internet, select **Use a Proxy Server**. Then fill in the rest of the form.

Ask your system administrator for the host address, port assignment, and proxy type. You may also find the proxy information in the options or preferences of your browser. Check for Advanced or Connections options.

6. Click **OK**.

Getting a Service Patch

There are two ways to get service patches: checking manually with the UPDATE program or, more conveniently, through the Start Page of Diamond's main window. If you enabled automatic checking in the UPDATE options, Diamond notifies you whenever a new service patch becomes available.

In Windows, if a service patch is available, you have a choice between installing the patch immediately or downloading it now and installing it later, at a convenient time.

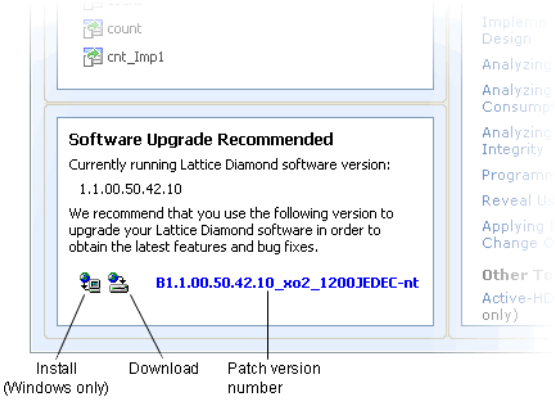
In Linux, you download the service patch and install it manually at your convenience.

Whether you are installing immediately or just downloading the service patch, getting it may take several minutes. Depending on the size of the patch, installing it may take a few more minutes.

To get a service patch from the Start Page:

1. If you do not have automatic checking enabled, choose **Help >** **Check for Updates**.
2. Check the lower-left corner of the Start Page. If there is a service patch available, the frame is titled "Software Upgrade Recommended" and shows a list of patches as shown below.

Figure 1: Part of Start Page Showing a Patch Available



3. For information about a patch, click the patch version number.

   Your default browser opens with an update notice. This notice includes information about installing the patch and links to the Lattice Web site for more information.

4. After reading the update notice, do one of the following:

   - (Windows only) Click the Install icon that is next to the version number to immediately install the service patch. Close all Diamond tools and then restart Diamond.
   - Click the Download icon that is next to the version number to save the service patch to a directory and install it later. In the Save File dialog box, navigate to the location where you want to save the file and click **Save**.

   Hover the cursor over an icon to see a label for it.

To get a service patch from UPDATE:

1. Start UPDATE:

   - In Windows: Choose **Start > Programs > Lattice Diamond > UPDATE**.
   - In Linux: On a command line, enter <*install_path*>**/bin/lin/update**.
2. If you have more than one version of Diamond installed, choose the version that you want to update in the “Installed versions” box.
3. If you do not have automatic checking enabled, click **Update**.
4. Check the “Available update versions” box. If there is a service patch available, the box shows a list of patches.
5. For information about a patch, click the patch version number.

   An update notice appears in the lower portion of the window. This notice includes information about installing the patch and links to the Lattice Web site for more information.
6. After reading the update notice, do one of the following:

   - (Windows only) Click **Install** to immediately install the service patch.
   - Click **Download** to save the service patch to a directory and install it later. In the Save File dialog box, navigate to a location where you want to save the file and click **Save**.

See Also

- “Setting Options for Updates” on page 17
- “Installing a Downloaded Service Patch” on page 19

Installing a Downloaded Service Patch

After you have downloaded a service patch, you can install it to update your Diamond software.

Installing on Windows To install the service patch:

1. Close all Diamond tools.
2. Go to the location where you saved the service patch.

3. Double-click the service patch's executable file (.exe).

   The Lattice Semiconductor Setup program opens.

4. Follow the on-screen instructions.

Installing on Linux To install a service patch, you need to be listed in the system's sudoer file. The sudo utility allows the files to be installed with you as the owner and with the proper permissions.

To install the service patch:

1. Close all Diamond tools.
2. On a command line, change to the directory holding the service patch:

   cd <*temp_tools*>

3. Run the service patch's executable file (.rpm) using the sudo and rpm utilities:

   sudo rpm -ivh <*service_patch*>**.rpm**

Copyright, Trademarks, and Disclaimers

Copyright Copyright © 2020 Lattice Semiconductor Corporation. All rights reserved. This document may not, in whole or part, be reproduced, modified, distributed, or publicly displayed without prior written consent from Lattice Semiconductor Corporation ("Lattice").

Trademarks All Lattice trademarks are as listed at www.latticesemi.com/legal. Synopsys and Synplify Pro are trademarks of Synopsys, Inc. Aldec and Active-HDL are trademarks of Aldec, Inc. Modelsim and Questa are trademarks or registered trademarks of Siemens Industry Software Inc. or its subsidiaries in the United States or other countries. All other trademarks are the property of their respective owners.

Disclaimers NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF LATTICE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Lattice may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. Lattice makes no commitment to update this documentation. Lattice reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. Lattice recommends its customers obtain the latest version of the relevant information to establish that the information being relied upon is current and before ordering any products.

Chapter 2

Managing Projects

Managing design projects involves a variety of activities: creating and maintaining the project, keeping track of the stages in the design implementation process, reviewing reports detailing the results of the process, and comparing different versions (implementations) of the project. Managing projects can also include writing Tcl scripts to automate your own design and test procedures.

Creating a project in Diamond is a simple matter of specifying a location for the project files, any existing design source files, and the FPGA type. You do this with the New Project wizard (choose **File > New > Project**). Diamond creates a folder with a project file (.ldf) containing basic information about the project, the beginning of a logical preference file (.lpf), which controls how the design is implemented in the map and place-and-route stages, and a default strategy (Strategy1.sty).

A “strategy” is a collection of settings for controlling the different stages of the implementation process (synthesis, map, place & route, and so on). Strategies can control whether the design is optimized for area or speed, how long place and route takes, and many other factors. Diamond provides a default strategy, which may be a good collection to start with, and some variations that you can try. You can modify Strategy1 and create other strategies to experiment with or to use in different circumstances.

The project folder also contains an implementation folder. Implementation folders contain the source files, process reports, and other information for different implementations, or versions, of a design. Having different project implementations helps you to experiment and compare different designs. With Run Manager (choose **Tools > Run Manager**) you can process multiple implementations simultaneously and compare the results. You can check the details of each implementation in a set of process reports (choose **View > Reports**).

As you develop your design, routine tasks can be controlled through the Diamond graphical user interface or through scripts. Diamond comes with command-line and Tcl commands for many of its functions.

See Also ▶“Creating a New Project” on page 24

- “Using Strategies” on page 48
- “Working with Implementations” on page 46

▶ "Viewing Logs and Reports" on page 101

Running Lattice Diamond

The Diamond main window is the primary interface and provides an integrated environment for managing the project elements and processes, as well as accessing all Lattice Diamond tools and views.

To run the Diamond main window:

- From your Windows desktop, choose **Start > Programs > Lattice Diamond > Lattice Diamond**.

  Note

  Lattice Diamond is the default Programs folder name when you install the Lattice Diamond software. Change this name accordingly if you have chosen another folder name during installation.

- From your Linux platform shell window or C-shell window, execute:

  `<install_path>/bin/lin/diamond`

Pin Diamond to Start Menu or Taskbar

You can pin Diamond to your Windows Start menu or taskbar, so you can open it quickly and conveniently, rather than looking for Diamond in the Start menu.

To pin Diamond to the Start menu or Taskbar:

- Choose the Windows **Start** menu. Find the **Lattice Diamond** icon. Right-click the **Lattice Diamond** icon, and then click **Pin to Start Menu** or **Pin to Taskbar**.

Note

You should not pin Diamond to the Taskbar through the minimized icon on the taskbar while Diamond is running.

If you want to remove the pinned Diamond from the Start menu or the taskbar, right-click on a pinned **Lattice Diamond** icon from the Start menu or the taskbar, choose **Unpin from Start Menu** or **Unpin this program from taskbar**.

Managing Diamond Layout

The Diamond main window has many controls, views, tools, reports, outputs, and a Tcl console. They are very flexible and can be configured to store your preferred layout by using the Load Layout, Manage Layout, and Save Layout menu commands.

Refer to the Lattice Diamond User Guide, the “User Interface Operation” chapter, for more details.

Creating a New Project

A project is a collection of all files necessary to create and download your design to the selected device. The New Project wizard guides you through the steps of specifying project name and location, selecting a target device, and adding existing sources to the new project.

Note

Do not place more than one project in the same directory.

To create a new project:

1. From the Diamond main window, choose **File > New >** **Project**.

   The New Project wizard opens.

2. Click **Next**.

3. In the Project Name dialog box, do the following:

   - Under Project, specify the name for the new project.

     File names for Diamond projects and project source files must start with a letter (A-Z, a-z) and must contain only alphanumeric characters (A-Z, a-z, 0-9) and underscores (_).

   - To specify a location for your project, click **Browse**. In the Project Location dialog box, you can specify a desired location.

   - Under Implementation, specify the name for the first version of the project. You can have more than one version, or “implementation,” of the project to experiment with. For more information on implementations, refer to “Working with Implementations” on page 46.

   - When you finish, click **Next**.

4. In the Add Source dialog box, do the following if you have an existing source file that you want to add to the project. If there are no existing source files, click **Next**.

a. Click **Add Source**. You can import HDL, schematic, EDIF netlist, LPF constraints, or Reveal Inserter project files at this time.

Tip

When you create a new project, the Diamond software always creates a default <*project*>.lpf file for the project. If you already have an .lpf file and want to use it in your new project, you will need to add this .lpf into the project. In this way this pre-existing .lpf file will be used as the active .lpf. Otherwise, you will need to add this pre-existing .lpf later, and mark the <*project*>.lpf inactive.

b. In the Import File dialog box, browse for the source file you want to add, select it, and click **Open**.

The source file is then displayed in the Source files field.

c. Repeat the above to add more files.

d. To copy the added source files to the implementation directory, select **Copy source to implementation directory**. If you prefer to reference these files, clear this option.

e. When you finish, click **Next**.

5. In the Select Device dialog box, select a device family and a specific device within that family. Then choose the options you want for that device. When you finish, click **Next**.
6. In the Select Synthesis Tool dialog box, select the synthesis tool that you want to use. This choice can be changed at any time. See "Selecting a Synthesis Tool" on page 89 for more information. When you finish, click **Next**.
7. In the Project Information dialog box, make sure the project settings are correct and then click **Finish**.

Note

If you want to change some of the settings, click **Back** to modify them in the previous dialog boxes of the New Project Wizard.

See Also

- "Tips for Saving and Naming Projects" on page 32
- "Reserved File Names" on page 32

Modifying a Project

After creating a project, you can modify the project by using the right-click menu.

To modify a project:

1. Right-click on any part of the project in the File List view.

The right-click menu varies upon the different part of the project you have chosen.

2. You can choose to edit a device in the Device Selector, add source files, implementations, or strategies as needed to the project, clone a strategy, set the current highlighted strategy as active strategy, exclude certain source files from an implementation or remove a file.
3. You can also edit properties of the project in the Project Properties dialog box. You can set the top-level unit, specify the VHDL Library name, and Verilog Include Search path in the Project Properties dialog box.

Importing ispLEVER Projects

Design projects created in ispLEVER can easily be imported into Diamond. The process is automatic except for the ispLEVER process properties, which are similar to the Diamond strategy settings, and PCS modules. All your ispLEVER project settings can be handled smoothly in Diamond. After importing a project, you need to set up a strategy for it and regenerate any PCS modules.

Note

Before converting an ispLEVER project, make a backup copy of the project. In ispLEVER, open the project and choose **File > Archive Current Project**. See the ispLEVER Help for details.

To import an ispLEVER design project:

1. In Diamond, choose **File > Open > Import ispLEVER Project**.
2. In the Select ispLEVER Project dialog box, browse to the project's **.syn** file.
3. Click **Open**.

   The Import ispLEVER Project dialog box opens showing the .syn file.
4. If desired, change the base file name or location for the Diamond project. If you change the location, the new Diamond files will go into the new location. By default, the "Copy design source to Implementation's Source directory" option is checked. The original source files will be copied to the new location.

   Note

   If you choose to copy source files when importing an ispLEVER project, the following limitations need to be noted:

   - Only source list in the project will be copied to your current project directory.
   - The files, .v, .txt, .ngo, and so on, related to an LPC file listed in the project will not be copied. You can copy them to the current project directory manually.
   - Schematic symbols will not be copied.
   - Included Verilog files (specified in the source files) will not be copied.
5. Click **OK**.

The project files are converted to Diamond format with the default strategy settings.

6. Adjust the strategy settings.

See Also ▶"Adjusting PCS Modules" on page 27

Adjusting PCS Modules

PCS modules created with IPexpress have an unusual file structure and need additional adjustment when importing a project from ispLEVER. There are two ways to do this adjustment. The preferred method is to regenerate the module in Diamond. However this may upgrade the module to a more recent version. An upgrade is usually desirable but if, for some reason, you do not want to upgrade the PCS module, you can manually adjust the module by copying its .txt file into the implementation folder. If you use this method, you must remember to copy the .txt file into any future implementation folders.

To regenerate a PCS module:

1. Find the module in the Input Files folder of the File List view. The module may be represented by an .lpc, .v, or .vhd file such as pcs_1.lpc.
2. If the File List view shows the Verilog or VHDL file for the module, and you want to regenerate the module, import the module's .lpc file:
   a. In the File List view, right-click the implementation folder  and choose **Add > Existing File**.
   b. Browse for the module's .lpc file, <*module_name*>**.lpc**, and select it.
   c. Click **Add**.

      The .lpc file is added to the File List view.
   d. Right-click the module's Verilog or VHDL file and choose **Remove**.
3. In File List, double-click the module's .lpc file.

   The module's dialog box opens.
4. In the bottom of the dialog box, click **Generate**.

   The Generate Log tab is displayed. Check for errors.
5. Click **Close**.

   In File List, the .lpc file is replaced with an .ipx file. The IPexpress manifest (.ipx) file is new with Diamond. The .ipx file keeps track of the files needed for complex modules.

To manually adjust a PCS module:

1. Open a file window and browse to the new Diamond project.
2. Find the module's .txt file, <*module_name*>**.txt**.
3. Copy the .txt file into the Diamond project's implementation folder. The implementation folder's name is based on the name of the .syn file. For

example, if the ispLEVER project has a design_pcs.syn file and a PCS module named pcs_1, move pcs_1.txt into the design_pcs folder.

Note

If you choose to copy source files when importing an ispLEVER project, the following limitations need to be noted:

- Only source list in the project will be copied to your current project directory.
- The files, .v, .txt, .ngo, and so on, related to an LPC file listed in the project will not be copied. You can copy them to the current project directory manually.
- Schematic symbols will not be copied.
- Included Verilog files (specified in the source files) will not be copied.

See Also "Importing ispLEVER Projects" on page 26

Targeting a Device

Lattice Diamond lets you retarget a design to a different device any time during the design process.

To target a device:

1. In the File List view, double-click the device name.

   The Device Selector dialog box opens. It contains all available devices and their options.

2. Under Select Device, select a device family and a specific device within that family. Then choose the options you want for that device.

   You can also click **Online Data Sheet for Device** to view the data sheet for any specific device.

   Note

   If you could not find the device you want in the device list, select **Show Obsolete Devices**. Then the device list will also include obsolete devices.

3. When you finish, click **OK**.

   The specified target device appears in the File List view.

Note

If you have an .ipx file whose device is not the same as the project device, double-click the .ipx file in the File List view. You will get a Mismatch Device message asking you to run IPexpress to regenerate the .ipx file. Click **Yes** if you want to regenerate the .ipx file.

Viewing Project Properties

After creating a project, you can view the project-related information in the Project Properties dialog box.

To view project properties:

1. Right-click on any part of the project in the File List view, and choose **Properties**.

   The Project Properties dialog box opens.

2. In the dialog box, you can see the file name, category, and location of the selected file. You can even enter values for some of the selected sections of the project in the Value field.

Saving Project Files

You can save changes to source files and to project properties. You can also save copies of individual files and whole design projects.

Saving Changes

As you are making changes to source files or to the project properties you should frequently save the files. Diamond offers three save commands for this:

- **File > Save**: Saves the currently active item.
- **File > Save All**: Saves all the content changes in the Power Calculator files (.pcf), Reveal Project files (.rvl), schematic files (.sch), Verilog files (.v, .veri, .ver, .vo, .h), VHDL files (.vhd, .vhdl, .hdl, .vho), EDIF netlist files (.edf, .edif, .edi, .edn, .edj), preference files (.lpf, .prf, .tpf), text files (.txt), synthesis constraint files (.ldc, .sdc), timing preference files (.tpf), waveform files (.wdl), memory files (.mem), Reveal Analyzer files (.rva), symbol library files (.lib), symbol files (.sym), and manifest files (.ipx).
- **File > Save Project**: Saves project properties such as the target device, implementation file lists, and strategy setting.
- **File > Save Project As**: Opens the Save Project As dialog to save the active project.

Also, if there are any unsaved changes when you close a project, a dialog box will open offering a chance to specify which files you want to save.

Copying a Source File

You can create copies of source files using Diamond.

If you make a copy of a source file in the same project folder as the original file, remember that they must have different names and that modules in different files must have different names.

To copy a source file:

1. Open the source file with a Diamond tool.
2. Choose **File > Save** <*filename*> **As**.

   The “Save as” dialog box opens.
3. Browse to where you want to save the file.
4. Enter a new file name.

   Note

   In the Linux version of Diamond, when you save or export a file, the dialog box does not automatically append an extension to the file name you specify. That is the standard way Linux handles the **Save As** dialog box. You need to manually add the relevant file extension.
5. Choose the file type from the “Save as type” menu. Be careful to choose the right type or your copy will have the wrong extension and tools will not recognize it.
6. Click **Save**.

Copying a Project

You can make a copy of a design project to use as the beginning of another project. (If you want the copy to backup or move the project, see “Archiving a Project” on page 31 instead.) This process saves all source and other project files that are within the project folder to another folder. Referenced files are not included.

To copy a design project:

1. Choose **File > Save Project As**.

   The Save Project As dialog box opens.
2. In the dialog box, browse to where you want to save the copy.
3. Click the Create New Folder  button to create a new project folder.
4. Type a new name for the folder and press **Enter**.
5. Double-click the new folder to open it.
6. If desired, give the project a new name in the “File name” box.
7. If you want the copy to include files generated by the implementation processes, select **Copy generated files**.
8. Click **Save**.

   If there are open files, the Save Modified Files dialog box may open. Select files to be saved before copying and click **OK**.

The current design project closes and the new one opens.

Check references to files outside of the project folder. These may need adjustment.

See Also ▶"Tips for Saving and Naming Projects" on page 32

Archiving a Project

Diamond offers you an easy way to archive the current project. A project archive is a single compressed file (.zip) containing the information for the entire project. After archiving a project, you can reload it in the main window at any time.

When you archive a project that contains source files stored outside the project folder, the remote files are compressed under the remote_files subdirectory in the archive.

To archive the current project:

1. In the main window, make sure the project you want to archive is open.
2. Choose **File > Archive Project**.
3. In the Archive Project dialog box, specify the file name and location.
4. If you want to include all files in the project directory, not just source files, select **Archive all files under the Project directory**.

   Clearing this option may make a much smaller archive by leaving out nonessential files such as temporary files or documentation. Without this option Diamond archives only those files it recognizes as part of the project. *With* this option Diamond archives *all* files in the project directory of all kinds.
5. Click **Save**.

   The archive file is created and stored in the specified location.

To reload the archived project:

1. In the Diamond main window, choose **File > Open > Archived Project**.
2. In the Select Archived Project File dialog box, browse to the archive file.
3. Click **Open**.
4. In the Open Archived Project dialog box, check the destination directory. If this is not where you want to place the project files, click the Browse (**...**) button.
5. In the Browse for Folder dialog box, browse to where you want to place the project files.
6. Click **OK**.
7. In the Open Archived Project dialog box, click **OK**.

The project files are extracted to the specified folder and the project is opened in Diamond.

See Also ▶"Copying a Project" on page 30

Tips for Defining Projects

Use the following guidelines when defining your projects:

- Avoid using VHDL, Verilog HDL, or EDIF reserved words for module and signal names in any of your source files.
- Each source file must have a unique name in the project. You cannot have two different source files with the same name. You can use the same source file many times in a design by instantiating the source file. Two different source files with the same name can cause problems with the hierarchy. For example, do not use a Verilog file called "compare" and a schematic file also called "compare."
- Do not include the same module name in different sources within one project.

Tips for Saving and Naming Projects

Use the following guidelines when saving and naming source files and your projects:

- Do not save more than one project in the same directory.
- File names for Lattice Diamond projects and project source files must start with a letter (A-Z, a-z) and must contain only alphanumeric characters (A-Z, a-z, 0-9) and underscores (_).
- Try to shorten the length of the path name and levels of the directory. Problems might occur when the path of a project is too long.

Reserved File Names

Lattice Diamond reserves some filename extensions for its own use. You should avoid using the following extensions when naming your own files:

File Extension	File Use
_sc	Schematic Editor log file
_sy	Symbol Editor log file
_wt	Waveform Editor Viewer log file
_wv	Waveform Viewer log file
.asc	ASCII schematic file

File Extension	File Use
.asy	ASCII symbol file
.ed*	EDIF netlist
.err	Error output file
.fp0, .fp1	ORCAstra configuration file
.fpm	ORCAstra macro file
.his	Waveform Viewer history file
.ipx	Manifest file. This file is loaded into IPexpress so that modifications can be made to the module.
.lci	Constraint file
.lco	Constraint output from the Fitter (such as the post-fit pinouts)
.lct	Temporary working copy of the constraint file
.ldc	Lattice synthesis constraint file
.ldf	Lattice Diamond project file
.lpc	Lattice parameter configuration file created with IPexpress in ispLEVER
.lpf	Logical preference file
.nam	Binary waveform name file
.pcf	Power Calculator project file
.pin	Netlist file for generic netlist by pin
.rva	Reveal Analyzer file
.rvl, .rvs	Reveal Inserter project file
.sch	Schematic Editor files
.sty	Lattice Diamond strategy file
.sym	Symbol Editor file
.syn	ispLEVER project file (reserved only when importing an ispLEVER project into Lattice Diamond)
.spf	Lattice Diamond Simulation Wizard file
.tpf	Lattice Diamond Simulation timing preference file
.vis	ORCAstra custom Visual Window definition
.wav	Waveform Viewer waveforms and trigger information
.wdl	Waveform Editor display file
.wet	Waveform Editor database

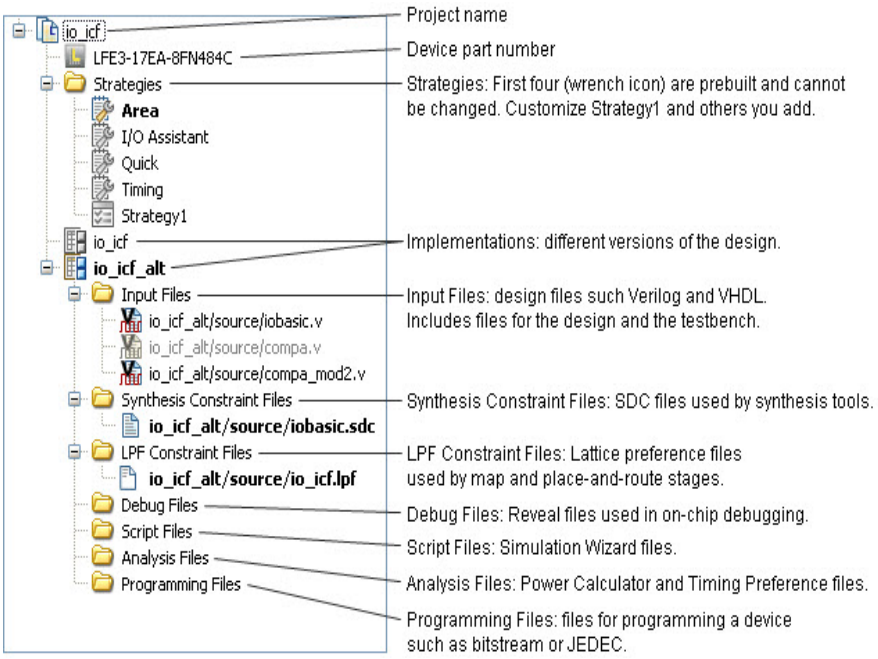
Managing Project Sources

The Diamond software combines design sources into different categories and list them in the File List view. You can see folders in the File List view. The source files are classified and listed under these folders. There are Strategies, Input Files, Synthesis Constraint Files, LPF Constraint Files, Debug Files, Script Files, Analysis Files, and Programming Files.

The HDL, schematic, edif netlist, and IP Module files can be found in the Input Files folder. The logic preference files are in the LPF Constraint Files folder. The Reveal project files can be found in the Debug Files folder. The Power Calculator, and timing preference files are in the Analysis Files folder.

You can also see implementations listed in the File List view. For details about implementations, see "Working with Implementations" on page 46.

File List View The File List view of the Diamond main window lists all the design sources. The sources are categorized by different types and are identified with different icons. Bold items are active and will be used when processing the design project. Grayed out items will not be used.



Files Listed in the File List View

File Type	Icon	File Extension
Project Title		None
Target Device		None
Predefined Strategy (Area, I/O Assistant, Quick, and Timing)		.sty

File Type	Icon	File Extension
Strategy1 (that can be customized)		.sty
Implementation		None
Verilog Files		.v, .veri, .ver, .vo, .h
SystemVerilog		.sv
VHDL Files		.vhd, .vhdl, .hdl, .vho
Schematic Files		.sch
EDIF Netlist Files		.edf, .edif, .edi, .edn, .edj
IP Module Config Files		.ipx
Undefined or incorrect		Any source reference
Synthesis Constraint Files		.sdc, .fdc, .ldc
Preference Files		.lpf
Reveal Project File		.rvl
Simulation Project File		.spf
Reveal Analyzer Files		.rva
Timing Preference File		.tpf
Power Calculator Files		.pcf

Note for Linux

In the Linux version of Diamond, when you save or export a file, the dialog box does not automatically append an extension to the file name you specify. That is the standard way Linux handles the Save As dialog box. You need to manually add the relevant file extension.

SystemVerilog Limitations The following tools do not support SystemVerilog files:

- Lattice Synthesis Engine (LSE)—Use Synplify Pro for Lattice or another synthesis tool that supports SystemVerilog.
- Hierarchy
- Simulation Wizard
- Reveal Inserter and Reveal Analyzer

See Also ▶"Running Processes" on page 70

Creating a New Source File

You can create a new source from within Diamond or external to Diamond.

To create a new source file:

1. In the Lattice Diamond main window, choose **File > New > File**.

   The New File dialog box opens.

2. In the dialog box, select the type of file you want to create. Fill in the file name, browse to choose a location for the file, check the **Add to project** option, choose the Implementation you want the file to be part of, and then click **New**.

   Diamond starts an editor that you can use to enter the information for your new source file. For language-based sources, the Source Editor is started. For schematic sources, the Schematic Editor is started. For module sources, IPexpress is started.

3. In the editor, create a source file.

   Some of the files can be directly added to the input files or constraint files folder of the File List view in Diamond, while some can be called by certain modules when running certain processes.

Note

- File names for Diamond projects and project source files must start with a letter (A-Z, a-z) and must contain only alphanumeric characters (A-Z, a-z, 0-9) and underscores (_).
- In the Linux version of Diamond, when you save or export a file, the dialog box does not automatically append an extension to the file name you specify. That is the standard way Linux handles the Save As dialog box. You need to manually add the relevant file extension.

See Also ▶"Analyzing a Design" on page 54

Importing an Existing Source File into a Project

You can easily import existing source files into your project. Source files for a project can be stored in different locations.

The files added to the project appear in the Input Files folder. You can adjust the file order by drag-and-drop. If several modules are detected as being uninstantiated, the last one will be automatically treated as the top-level module. You can also set the top-level module from the implementation property dialog box.

To import an existing source file:

1. Choose **File > Add > Existing File** from Lattice Diamond. Or, with a project opened in Lattice Diamond, right-click the current implementation and choose **Add > Existing File**.

   The Add Existing File dialog box opens.

2. Browse for the source file you want to import.
3. Select the file and click **Add**.

   The selected file is added to the Input Files folder of the File List view. If the source file is stored outside the project folder, the path of the file is displayed.

   Double-click the imported file. The file can be opened in the associated editor (this is a Windows-only feature).

Tip

- If you want a source file that is outside the project folder to be copied to the project directory, select the **Copy file to Implementation's Source directory when adding existing file** option from the Options dialog box, in the **Environment > General** section.
- Diamond project creation supports mixed language source files, allowing you to freely mix Verilog HDL, VHDL, and schematic design. The Diamond software will figure out the hierarchy structure that is associated with the module names. You need to pay special attention to the module names and the case, as Verilog is case-sensitive and VHDL is case-insensitive.

See Also "Analyzing a Design" on page 54

"Setting the Top-Level Unit for Your Project" on page 43

Importing an IPexpress Module

After generating module source files using IPexpress, you can import the IPexpress manifest file (.ipx) into Lattice Diamond. If the module was originally customized with ispLEVER, you may see an .lpc file but no .ipx file. The .ipx file is new with Diamond. However, Diamond also supports .lpc files for backward compatibility. The ispLEVER .lpc file can be converted to a Diamond .ipx file by re-generating the IPexpress module. When you regenerate a module from Diamond, the .lpc file is replaced with an .ipx file. For details on how to import ispLEVER modules, refer to "Adjusting PCS Modules" on page 27.

To import a module source (.ipx) file:

1. In the Lattice Diamond main window, choose **File > Add > Existing File**.

   The Add Existing File dialog box opens.

2. Browse for the .ipx file you generated and select it.
3. Click **Open**.

The selected .ipx file is added and listed in the File List view.

Note

When you import an existing ispLEVER project with an existing .lpc file, you need to re-open IPexpress using the .lpc file, regenerate the module, then add the .ipx file to the Lattice Diamond project.

For information on how to create a module source file, refer to "Designing with Modules" on page 190.

See Also ▶"Modifying a Source File in IPexpress" on page 45

Importing an IPexpress IP Core

After generating LatticeCORE IP source files using IPexpress, you can import the IPexpress manifest file (.ipx) into the Lattice Diamond main window. If the module was originally customized with ispLEVER, you may see an .lpc file but no .ipx file. The .ipx file is new with Diamond. However, Diamond also supports .lpc files for backward compatibility. When you regenerate an IP core from Diamond, the .lpc file is replaced with an .ipx file. For details on how to import ispLEVER modules, refer to the "Adjusting PCS Modules" on page 27.

To import an IP source (.ipx) file:

1. In the Lattice Diamond main window, choose **File > Add > Existing File**.

   The Add Existing File dialog box opens.

2. Browse for the .ipx file you generated and select it.
3. Click **Open**.

   The selected .ipx file is listed in the File List view.

Note

Some IP cores require factory assistance to generate custom configurations. See the Intellectual Property section of the Lattice Web site for more details on which IP cores are user-configurable versus factory-configurable.

For information on how to create an IP core, refer to "Designing with Modules" on page 190.

See Also ▶"Modifying a Source File in IPexpress" on page 45

Adding Reveal Debug Information

Reveal Inserter manages the addition of Reveal debug information into the Lattice Diamond source file. By default the Reveal Inserter project file (.rvl) will be automatically added to the File List view once Reveal Inserter has successfully merged trigger and trace signal features into your design. The procedure below describes how to manually import an .rvl file.

To import a Reveal project file (.rvl):

1. In the Lattice Diamond main window, choose **File > Add > Existing File**.

   The Add Existing File dialog box opens.

2. Browse for the .rvl file you generated and select it.
3. Click **Add**.

   The selected .rvl file is listed in the File List view.

   You can have multiple Reveal project files for your project. However, you can only set only one .rvl file active at one time.

For more information, refer to "Testing and Debugging On-Chip" on page 1040.

Adding a LatticeMico Platform

LatticeMico platforms are modeled with a behavioral Verilog HDL file (<*platform_name*>.v) and in some cases a combination of IP core peripherals generated by IPexpress, and custom peripherals written in VHDL. The Mico System Builder (MSB) creates a design file list of the entire platform to ease importing into the Lattice Diamond main window.

To import a platform design file list output by MSB:

1. In the Diamond main window, choose **Tools > Options**.
2. In the Options dialog box, expand the Environment list and select **General**.
3. Make sure that the **Copy file to Implementation's Source directory when adding existing file** option is cleared, and click **OK**.
4. In the File List pane of Diamond, create a new implementation, or select the implementation that you want to use for the platform files.
5. Choose **File > Add Existing File**.
6. In the dialog box, browse to the MSB platform file list <*platform_name*>soc location, select the source files to be added, and click **Add**.
7. In the Choose Top Module dialog box, select the top-level source file from the drop-down menu and click **OK**.

For complete information on creating a microprocessor platform, refer to the Lattice Mico System Builder section of the LatticeMico System online Help.

Importing Test Stimulus Files Using Simulation Wizard

If your design needs Verilog test fixture (.v) or VHDL test bench (.vhd), use the **Simulation Wizard**.

If multiple hierarchical test stimulus files are to be used, you should first import the top-level test file, and then add lower-level test stimulus as dependency files. Lattice Diamond will run all these test files in a hierarchical order during the simulation process.

To import a test stimulus file:

1. In Lattice Diamond, choose **Tools > Simulation Wizard**.

   The Preparing the Simulator Interface dialog box opens.

2. Click **Next**.
3. In the Simulator Project Name dialog box, enter a name for your simulation project in the Project name field. Browse to find a desired location for your project. Then choose either Active-HDL or ModelSim as the Simulator.
4. In the Process Stage dialog box, all the available process stages of the FPGA implementation strategy are automatically displayed. Choose the stage you want to run simulation.
5. Click **Next**.
6. In the Add Source dialog box, browse for your desired Verilog test fixture (.v) or VHDL test bench (.vhd) and select it. Check the **Copy Source to Simulation Directory** option if you need.
7. Click **Next**.
8. In the Summary dialog box, all your simulation project information is listed, including simulator name, project name, project location, simulation stage, simulation files, and simulation libraries.

   Check the “Run simulator” option if you want to run the simulation right away.

9. Click **Finish** to exit the Simulator Wizard. Your simulation project (.spf) is added to the Script Files folder of the File List view.

Note

The simulation flow in Diamond 1.3 or later has been enhanced to support source files that can be set in the File List view to be used for the following purposes:

- Simulation & Synthesis (default)
- Simulation only
- Synthesis only.

This allows the use of test benches, including multiple file test benches. Additionally, multiple representations of the same module can be supported, such as one for simulation only and one for synthesis only.

Refer to “Simulating the Design” on page 310 for more details on how to use the Simulation Wizard.

Managing Constraint Files

There are different types of constraint files in Diamond.

- Synthesis constraint file: SDC, FDC, LDC
- LPF constraint file: LPF

About Synthesis Constraint Files Depending on the synthesis tool you chose, you can add .fdc files for Synplify Pro, .sdc files for Precision, or .ldc files for Lattice Synthesis Engine (LSE). The files are listed in the Synthesis Constraint Files folder in the File List view.

Adding, Activating, Editing, and Removing Synthesis Constraint Files You can add, activate, edit, or remove synthesis constraint files of your project.

To add a synthesis constraint file:

1. Right-click the **Synthesis Constraint Files** folder in the File List view, and choose **Add**.
2. Choose **New File** if you want to add a new synthesis constraint file. Choose **Existing File** if you want to add an existing synthesis constraint file.

   The selected file is added to your project.

To activate a synthesis constraint file:

- Right-click the synthesis constraint file in the File List view of Diamond, and choose **Set as Active**.

  The selected synthesis constraint file is activated.

Note

- You can make zero or one .ldc constraint file active in your project.
- You can make multiple .sdc and .fdc constraint files active in your project.

To edit a synthesis constraint file:

- Double-click the synthesis constraint file you want to edit from the File List view of Diamond. Or, right-click the synthesis constraint file you want to edit from the File List view of Diamond.

  The selected synthesis constraint file is opened in the Source Editor. You can edit the selected file in the editor. See “Using Templates” on page 171 for more information.

To remove a synthesis constraint file:

- Right-click the synthesis constraint file in the File List view of Diamond, and choose **Remove**.

  The selected file is removed from your project.

About LPF Constraint Files You can add one or multiple logic preference files (.lpf) to your project. They are listed in the LPF Constraint File folder of the File List view.

Note

You can set only one .lpf file active in your project at one time.

Adding, Activating, Editing, and Removing LPF Constraint Files You can add, activate, edit, or remove LPF constraint files (.lpf) of your project.

To add an LPF constraint file:

1. Right-click the LPF Constraint Files folder in the File List view of Diamond, and choose **Add**.
2. Choose **New File** if you want to add a new LPF constraint file. Choose **Existing File** if you want to add an existing LPF constraint file.

   The selected file is added to your project.

To activate an LPF constraint file:

- Right-click the LPF constraint file in the File List view of Diamond, and choose **Set as Active**.

  The selected file is activated.

Note

You can set only one .lpf file active in your project at one time.

To edit an LPF constraint file:

- Double-click the LPF constraint file you want to edit from the File List view of Diamond. Or, right-click the LPF constraint file you want to edit from the File List view of Diamond.

  The selected LPF constraint file is opened in Source Editor. You can edit the selected file in the editor. See "Using Templates" on page 171 for more information.

To remove an LPF constraint file:

- Right-click the LPF constraint file in the File List view of Diamond, and choose **Remove**.

  The selected file is removed from your project.

See "Applying Design Constraints" on page 360 for more information.

Setting the Top-Level Unit for Your Project

It is a good practice to specify the top-level unit (or module) of the design. If you don't, Diamond will try to determine the top-level unit. While usually accurate, there is no guarantee that Diamond will get the correct unit.

You may also want to change the top-level unit when experimenting with different designs or switching between simulation and synthesis.

Notes for EDIF Modules

- In a mixed language design, you cannot set an EDIF module as the top-level unit. This is also true when a project with EDIF modules is mixed with .ipx files, which represent modules and IP generated with IPexpress.
- In a pure EDIF design, do not use the following procedure. Instead, in the File List view, arrange the input files so that the top-level unit is in the last file in the list.

To set the top-level unit for your project:

1. In the File List view, right-click the active implementation folder  .
2. From the drop-down menu, choose **Properties**.

   The Project Properties dialog box opens.
3. Click in the Value cell for **Top-Level Unit**.
4. Type the name of the top-level unit.

   At its right end, the Value cell also shows an arrow for a drop-down menu. Usually the menu shows one name or just "<Edit>." But if Diamond found multiple possible top-level units, the menu will show those names. If the correct top-level unit is in the menu, you can choose it.
5. Click **OK**.

   If the hierarchy is up to date, the source file that contains the top-level unit is displayed in bold.

Modifying a Source File

You can edit any of the source files that make up your project by double-clicking on them to open the corresponding editor.

Note

In order for Windows file associations to work properly with Diamond, in the main window choose **Tools > Options**. Then in the Options dialog box, find the **Environment > File Associations** section.

Modifying a Source File in Schematic Editor Use the Lattice Diamond Schematic Editor to edit a schematic file (.sch). You can open the Schematic Editor in the following ways:

- Double-click the schematic file name from the File List view of Lattice Diamond.
- In the File List view, right-click the schematic file and choose **Open** or **Open with** from the pop-up menu.

The Symbol Editor works in conjunction with the Schematic Editor. You can open the Symbol Editor by creating a new a symbol file in Lattice Diamond. For details on how to create a symbol, see "Creating Schematic Symbols" on page 270 in the Schematic Design Entry Help for more details.

Modifying a Source File in Source Editor You can use the Source Editor to enter or edit VHDL (.vhd), Verilog HDL (.v), or Preference (.lpf) files.

You can open the Source Editor in these ways:

- Double-click the text file name in the File List view of Lattice Diamond.
- In the File List view, right-click the text file and choose **Open** or **Open with** from the pop-up menu.

Modifying a Source File in Your Favorite Text Editor Similarly, you can use any text editor to edit the above-mentioned files, and then import them into your project using **File > Add > Existing File** from the Lattice Diamond main window.

Note

A preference file (.lpf) can also be created or edited in the Spreadsheet View.

You can open any text editor in this way:

- In the File List view, right-click the text file and choose **Open with** from the pop-up menu. In the Open with dialog box, click **Add** if you want to add your favorite text editor; or, click **Edit**, if you want to edit your favorite text editor. In the Add or Edit dialog box, specify command arguments for the external editor. The mapping between the argument substitution values and the value which replaces them are:

  %F = File Name; %L = Line Number

  For example, to configure the gVim text editor, use the following command line: <*path*>/gvim +%L %F
- In the Options dialog box, choose **Options > Environment > File Associations**. From the File Associations table, you can see the file extension and the default associated program. You can choose the file type you want to open/edit with your favorite text editor. Click the **Add** button from the **External programs for ... file extension** field to add your favorite text editor. Use the **Edit** or **Remove** button to manage your file and associated editor list.

After adding or editing the external text editor, you can save it as your default text editor in the Open with dialog by clicking **Save As Default**.

Note

If you change a source file in a third-party text editor, you need to choose **Design > Refresh Design** so that Lattice Diamond can refresh the process list to reflect the current process status and update views such as Hierarchy.

Modifying a Source File in IPexpress You can use IPexpress to edit the module or IP core source (.ipx) file.

To modify a module or IP core source:

1. In the File List view, double-click the .ipx file.

   IPexpress opens.

2. Make modifications as you want, and then exit IPexpress.

The .ipx file in the File List view is updated. You may need to rerun the design to reflect the changes you made.

For information on how to use IPexpress, refer to "Designing with Modules" on page 190.

See Also ▶"Importing an IPexpress Module" on page 37

▶ "Importing an IPexpress IP Core" on page 38

Excluding a Source File

Sometimes you may want to exclude specific source files from logic synthesis and simulation. The excluded source files remain in your design project, but they will not be synthesized or simulated by the software.

To exclude a source file:

1. In the File List view, right-click the target source file and choose **Exclude from Implementation**.

   The selected file is excluded from the implementation. The target file is grayed out.

Note

To include the source file again, right-click the target source file and choose **Include in Implementation**.

Removing a Source File

You can remove a source from a project by using the Lattice Diamond **Remove** command, or the Windows' **Delete** command.

Removed source files are no longer part of your project. If you just want to exclude a file from logic synthesis and simulation, see "Excluding a Source File" on page 45.

To remove a source file from a project:

1. In the File List view, select the file that you want to remove.
2. Right-click the filename and choose **Remove**, or press the Delete key on your keyboard.

   The Lattice Diamond software removes the source from the project.

Working with Implementations

Implementations define the design structural elements for a project. An implementation contains the structure of a design and can be thought of as the source and constraints to create the design. For example, one implementation may use inferred memory and another implementation may use instantiated memory. There can be multiple implementations in a project, but only one implementation can be active at a time and there must be at least one implementation.

An implementation is automatically created whenever you create a new project. You can create new implementations using the source files of an existing implementation. You can also clone (copy) an existing implementation including all the related files and settings.

Implementations consist of:

- Input files
- Constraint files
- Debug files
- Script files
- Analysis files

Creating New Implementations

To create a new implementation:

1. With the project opened in Lattice Diamond, choose **File > New > Implementation**.
2. In the New Implementation dialog box, enter a name for the new implementation.

This name also becomes the default name for the folder of the implementation.

3. Change the name of the implementation's folder in the Directory text box if desired.
4. Change the location of the implementation's folder if desired.
5. Choose the synthesis tool from the drop-down menu.
6. Choose the default strategy from the drop-down menu.
7. Add sources files by clicking **Add Source** and choosing either:
   - **Browser** to select individual files.
   - **From Existing Implementation >** <*implementation name*> to use the source files of an implementation that already exists in this design project.

   Remove unwanted files by selecting the files and clicking **Remove Source**.
8. If you want to copy all of the source files into the new implementation folder, giving you the freedom to change the files without affecting other implementations, select the **Copy source to implementation directory** option.
9. To view or set properties of the source files, select a file and click **Properties**. The Properties dialog box opens. To change a property, click in its **Value** cell. Enter or choose the new setting. Then click **OK**.
10. Click **OK**.

Cloning Implementations

To clone an implementation:

1. In File List view, right-click on the name of the implementation that you want to copy and choose **Clone Implementation**.

   The Clone Implementation dialog box opens.
2. In the dialog box, enter a name for the new implementation. This name also becomes the default name for the folder of the implementation.
3. Change the name of the implementation's folder in the Directory text box if desired.
4. Decide how you want to handle files that are outside of the original implementation directory. Select one of the options:
   - **Continue to use the existing references**

     The same files will be used by both implementations.
   - **Copy files into new implementation source directory**

     The new implementation will have its own copies that can be changed without effecting the original implementation.
5. Click **OK**.

6. In File List view, right-click on the name of the default strategy and choose Edit.

   The Strategies dialog box opens.

7. In the Process column, choose **Translate Design**.
8. Check the search paths. If they point to the folder of the original implementation, change them to use the folder of the new implementation
9. Click **OK**.

Customizing Files in an Implementation After creating an implementation, you can choose to include or exclude source files for the implementation. You can customize files that are under the Constraint Files, Debug Files, Script Files, and Analysis Files folders.

Customizing Input Files You can choose to:

- Add a new source file or add an existing file.
- Include the file for Synthesis, Simulation, or Synthesis and Simulation.
- Exclude the file from the current implementation.
- Remove the file from the project.

Activate Files You can have multiple files under each of the Constraint Files, Debug Files, Script Files, and Analysis Files folders in the File List view. But only one file can be active in each of those folders. To change a file's status, right-click the file and choose **active** or **inactive**.

Saving an Implementation If you want to save an implementation to the project, choose **File > Save Project** from the Diamond main window.

See Also ▶"Using Strategies" on page 48

Using Strategies

A **strategy** provides a unified view of all settings related to the optimization controls of implementation tools, such as logic synthesis, mapping, and place and route. Any number of strategies can be applied to your project to find the best solution to meet your design objectives. For example, you might have one strategy with a very fast placement and routing attempt to get a quick look at the design, and another one with a higher effort level that's more thorough and takes longer to try to meet timing constraints. Although you can create an unlimited number of strategies, only one can be active at a time for each implementation, and each implementation must have an active strategy.

Strategy settings are listed in the Strategies dialog box. You can open the dialog box by double-clicking a strategy name in the File List view. Each strategy is stored as an .sty file.

There are two different kinds of strategies: predefined strategy and customized strategy.

Predefined Strategy Several strategies are predefined and supplied by Lattice: Area, I/O Assistant, Quick, and Timing. They are designed to solve particular types of designs.

- **Area**

  The Area strategy will try to minimize area by enabling the tight packing option available in map. Use this strategy to achieve area requirements. It is commonly used for low density devices such as MachXO and MachXO2.

  The benefit is that if it works, you will get results in area reduction. However, applying this strategy to large and dense designs may cause some difficulties in the place-and-route process with longer run time or not completing routing.

- **I/O Assistant**

  This I/O Assistant strategy will try to place I/O efficiently for your design at an early stage. It helps to check if you set a legal I/O location that affects board-level pin-outs.

  The benefit is that if it works, you will get results in I/O placement information at first without long runtime after finishing place and route. However, applying this strategy to your design may take extra runtime on the design, because it only executes logic synthesis, translation, map, and I/O placement processes.

  Note

  If you choose the I/O Assistant strategy for your project, the generated NCD file is incomplete. Running the Export Files > Bitstream File or Export Files > JEDEC File process (in the Process view) may fail.

  If you want to implement a complete design, you need to choose another strategy to rerun all the processes again.

- **Timing**

  The Timing strategy tries to achieve timing closure. The Timing strategy uses a very high effort level in place and route. Use this strategy if you are trying to reach the maximum frequency on your design. If you cannot meet timing requirement with this strategy, try to customize a strategy with individual settings.

  The benefit is that if it works, you will get results in good timing performance. However, this strategy may increase your runtime on place and route compared to the Quick and Area strategies.

- **Quick**

  The Quick strategy uses a very low effort level in place and route to get results with minimum processing time. If your design is small and your target frequencies are low, this is a good strategy. Even if your design is large, you might want to start with this strategy to get a first look at place-and-route results and to tune your preference file with minimum processing time.

The benefit is that if it works, you will get results in the least possible time. The quality of these results in terms of achieved frequency should be expected to be low, and large or dense designs may not complete routing.

The settings of predefined strategies cannot be modified in the Strategies dialog box. You can view their settings by:

- Double-clicking the predefined strategy name in the Files List view and opening the Strategies dialog box.
- Right-clicking on the predefined strategy name in the File List view and choosing **View** to open the Strategies dialog box.

Customized Strategy Excluding the predefined strategies, all other strategies can be customized in the Strategies dialog box. The settings of the strategy can be edited in the Strategy dialog box. You can edit its settings by:

- Double-clicking the strategy name in the File List view and open the Strategies dialog box.
- Right-clicking the strategy in the File List view and choosing **Edit** to open the Strategies dialog box.

See Also
- "Creating a Strategy" on page 50
- "Cloning a Strategy" on page 51
- "Adding an Existing Strategy" on page 51
- "Specifying Strategy Options" on page 52
- "Saving a Strategy" on page 53
- "Removing a Strategy" on page 53
- "Strategy Reference Guide" on page 1112

Creating a Strategy

You can create a new strategy for your design.

To create a new strategy in Lattice Diamond:

1. Choose **File > New > Strategy**.

   The New Strategy dialog box opens.

2. In the New Strategy dialog box, enter a name for the new strategy. Specify a file name for the new strategy and choose a directory to save the strategy file (.sty).

The new strategy is with all the default settings of the current design. You can modify its settings in the Strategies dialog box.

If you want to save the strategy changes to your current project, choose **File > Save Project** from the Diamond main window.

Cloning a Strategy

You can clone any strategy you created or any of the Lattice predefined strategies from the File List view. The new cloned file will contain all settings of the strategy being cloned. You can then modify the settings for the cloned strategy from within the Strategies dialog box.

To clone a strategy:

1. In the File List view, right-click on the strategy you want to clone and choose **Clone** <*name*> **Strategy**.
2. In the Clone Strategy dialog box, enter a name for the new strategy in the Strategy ID field. Specify a file name for the new strategy and choose a directory to save the strategy file (.sty). By default, when you enter the strategy ID, the file name is automatically specified the same.

The strategy you cloned is with all the settings of the base strategy. You can modify the settings for the cloned strategy in the Strategies dialog box.

If you want to save the strategy changes to your current project, choose **File > Save Project** from the Diamond main window.

Adding an Existing Strategy

You can add existing strategy to your design.

Note

A strategy copied from another design project may include inappropriate settings, such as path names, that could cause problems. If you copy a strategy from another project, review its settings carefully and adjust them as needed.

To add existing strategy to your design:

1. In Lattice Diamond, choose **File > Add > Existing Strategy**.
2. In the Select Existing Strategy dialog box, browse to the desired strategy file (.sty) and click **Open**.
3. In the Add Existing Strategy dialog box, enter the name for the added strategy as you want to see it appear in the File List view.
4. If the selected strategy file is not in the current project directory, you may want to select the **Copy strategy file to project directory** option.
5. Click **OK**.

   The new strategy appears in the File List view.
6. Double-click the new strategy to open the Strategies dialog box.
7. Carefully review the settings to make sure that they are appropriate for your current project. Be sure to check the following items:
   - Under Synthesize Design > LSE, check:

Macro Search Path

Memory Initial Value File Search Path

- Under Translate Design:

  Macro Search Path

  Memory Initial Value File Search Path

- Under Bitstream:

  Search Path

Note

If you did not select the “Copy strategy file to project directory” option when adding the strategy, remember that you are working with the original file, which might also be used by another project.

8. When your review is done, click **OK**.

If you want to save the strategy changes to your current project, choose **File > Save Project** from the Diamond main window.

Specifying Strategy Options

You can use the Strategy dialog box to specify settings for a specific design strategy.

To specify strategy settings:

1. In the File List view, double-click the active strategy.

   The Strategies dialog box opens.

2. In the left-hand pane, select a process for which you want to specify settings. The left-hand pane shows a process list. Each of the processes has its own settings.
3. In the right pane, double-click the Value column for the setting to be changed to activate the pulldown menu with choices of settings or to be able to enter filenames or directory paths into the field.
4. Select the value from the drop-down list, or type in the text value, and click **Apply**.
5. Repeat steps 2-4 to specify more settings.
6. When you finish, click **OK** to close the dialog box.

   All your changes to the strategy are updated in the system memory in your Diamond session. Be aware that these changes are not automatically written to your current project. If you want to save the

strategy changes to your current project, choose **File > Save Project** from the Diamond main window.

Tip

When a setting is highlighted, its brief definition will be displayed at the bottom of the dialog box. You can also press **F1** to view additional information for the highlighted option.

See Also ▶"Strategy Reference Guide" on page 1112

Saving a Strategy

If you want to save a new strategy, a cloned strategy, or any change of a strategy setting to your project, click **File > Save Project** from the Diamond main window.

Setting Active Strategy

If you have more than one strategies in the project, you can choose to set an active strategy.

To set an active strategy:

- Right-click the target strategy and choose **Set as Active Strategy**.

If you want to view the settings of the active Strategy, choose **Project > Active Strategy**. From the submenu, choose the settings you want to view. The Strategies dialog box opens with the settings you have chosen. You can view or edit the values of the settings.

Removing a Strategy

You can remove a strategy from the design by right-clicking on the target strategy and choosing **Remove**.

Getting Help on Strategies

To find the online Help description of strategy settings, do one of the following:

- Open the "Strategy Reference Guide" on page 1112 in the Contents pane of the Lattice Diamond Online Help. The Contents pane displays all process options in the order they appear in the Strategies dialog box. Choose the desired process option. You will see the strategies associated with this process. Use the scroll bar to select a setting and view its description.

- In the Lattice Diamond File List view, double-click a strategy name to open the Strategies dialog box. You can then highlight a setting to display its brief definition at the bottom of the dialog box, or press **F1** to view additional information for the highlighted option.

See Also "Strategy Reference Guide" on page 1112

- "Creating a Strategy" on page 50
- "Cloning a Strategy" on page 51
- "Adding an Existing Strategy" on page 51
- "Specifying Strategy Options" on page 52
- "Saving a Strategy" on page 53
- "Setting Active Strategy" on page 53
- "Removing a Strategy" on page 53

Analyzing a Design

There are a few ways to analyze your design while it is being implemented.

Before synthesis, the Hierarchy view provides a nested list of all modules plus commands to access the source code for individual modules, to set up test benches, and more. See "Hierarchy View" on page 54.

After synthesis, the Hierarchy view also provides information on how the modules are using device resources.

After synthesis, schematic views are also available. These views depend on which synthesis tool you are using. If you are using Lattice Synthesis Engine (LSE), you can use Diamond's Netlist Analyzer to see and analyze the design. If you are using Synplify Pro, you can use its HDL Analyst. See *Synplify Pro for Lattice User Guide.*

After mapping, the Hierarchy view provides updated information on how the modules are using device resources. Also, if you are using LSE, Netlist Analyzer shows how the design is mapped to the device's architecture.

Hierarchy View

The Hierarchy view shows the design hierarchy as a nested list of modules. The Hierarchy view opens automatically when you open a project. If the Hierarchy view is not open, choose **View > Show Views > Hierarchy** from the Diamond main window.

The Hierarchy view also shows the full path name of the files that define each module and, once synthesis has been run, the amount of device resources that each module would consume.

Commands in the Hierarchy View

Right-click in the Hierarchy view to get several useful commands described below.

Goto Source Definition This command opens the source editor with the source code that contains the definition for that object, and automatically scrolls the view to the position in the code where the object is defined. For objects that include multi-line definitions, the entire definition is highlighted.

Goto Source Instantiation This command brings you directly to the source file where the module is instantiated, making it easy to see and fix errors.

Goto RTL Definition After synthesis has been run with LSE, this command opens Netlist Analyzer with a schematic view of the design element.

Verilog Test Fixture Declarations This command creates the Verilog test fixture declarations include file (.tfi) based on the Verilog unit selected from the Hierarchy view. This include file should be referenced by your test fixture using: ‘include “<*file_name*>.tfi”. By including the.tfi file in your simulation test fixtures, you ensure that the design and the test fixtures stay synchronized.

Verilog Test Fixture Template This command creates a Verilog test fixture template file (.tft) based on the Verilog unit selected from the Hierarchy view. The template file includes a Verilog design unit with a module declaration. Use the procedure below to create a Verilog test fixture from the .tft file.

To create a Verilog test fixture from a Verilog Test Fixture Template (.tft) file:

1. Open the .tft file from the project directory and save it with a .v file extension.
2. Import the new .v file into the project as a Verilog test fixture. Then add codes to the user defined section to complete the stimulus for your design.

VHDL Test Bench Template This command creates a VHDL template file (.vht) based on the VHDL unit selected from the Hierarchy view. The template file includes a VHDL design unit with a component declaration and an instantiation based on the entity interface of the first VHDL design unit encountered.

Note

Avoid using user-defined record type array in your design. Try to use standard data type such as std_logic_vector. Using your own defined record type array may cause the signal width to be incorrect in the VHDL test bench template file generated.

Generate Schematic Symbol This command creates a schematic symbol (.sym) based on the design unit selected from the Hierarchy view. This process runs an independent utility to convert the NAF file that is associated with the selected design unit to a SYM file.

Note

Avoid using user-defined record type array in your design. Try to use standard data type such as std_logic_vector. Using your own defined record type array may cause the signal width to be incorrect in the schematic symbol file generated.

Set as Top-Level Unit You can directly set the selected module as the top-level unit by using this command.

Device Resources

Device resources that would be consumed are shown to the right of each module. The first number in each column is the number of that resource that would be used by that module and all of its submodules. The second number, in parentheses, is the number of that resource that would be used by just that module, not including any of its submodules. The numbers are updated after the synthesis and map stages of the implementation process. For definitions of the resource types, see Table 1 on page 56.

Note

If your source files for the design include one or more EDIF files, the resources shown may not represent the whole design. After synthesis but before the translate or map stages, results are only for the files that went through the synthesis tool. EDIF files, being the result of synthesis, are not included. After the translate or map stages, the results represent the whole design including any EDIF files.

The Hierarchy view only shows the design. It does not include any Reveal modules. However, when the top module of the design shows device resources, it does include the resources of any Reveal module. So, if you have included a Reveal module, the top module shows more resources than the submodules add up to.

Table 1: Resource Definitions

Hierarchy View Term	Synthesis Report Term	Definition
LUT4	OrcaLut, Lut	Four-input look-up tables that can implement Boolean functions with up to four variables
Registers	FD	Sequential elements with clock, enable, and synchronous or asynchronous set and reset
IO Registers	IF, OF	Sequential elements (registers) within input/output blocks

Table 1: Resource Definitions

Hierarchy View Term	Synthesis Report Term	Definition
DSP MULT	MULT	Configurable multipliers for high speed operations
DSP ALU	ALU	Configurable adders for high speed addition, subtraction, and logic functions like AND/OR
EBR	SP, DP, PDP	Embedded Block RAM to store large sets of data; configurable for word size and depth
Distributed RAM	SPR, DPR	RAM distributed with every logic element to store small sets of data such as register files
ROM	ROM	EBR configured as read-only memories
PFUMux	PFU	Dedicated multiplexers that implement 5-input functions using two LUT4 that directly drive the mux inputs
L6Mux	L6M	Dedicated multiplexers that implement 6-input functions using four LUT4 and two PFUMux that drive the L6Mux inputs
Carry cells	CCU	Dedicated logic cells to implement fast carry generation and propagation in adders
SLICE		Elements with two LUT4 that feed two registers and associated logic that can perform functions such as LUT8
Instantiated		Anything not covered by the other terms

Changing the Display

You can adjust the display by re-arranging the columns and sorting the rows. You can also choose which ones you want to see.

To change which columns are displayed:

1. Right-click in any column heading.
2. In the drop-down menu, the resource that you want to show or hide.

   Note: Resources that are not being used by the design at all are automatically hidden in Hierarchy view regardless of the option settings.

To re-arrange columns:

- Click on the heading of the column that you want to move and drag it to where you want it.

To sort the modules:

1. Click in the heading of the desired column.

   The rows are sorted according to the contents of that column.

2. To reverse the order of the rows, click in the same column again.

To change the width of columns:

- Click on the right-hand border of a column and drag to make that column narrower or wider.

Saving the Data You can save resource counts for the top-level module as either an ASCII (.txt) file or a comma-separated values (.csv) file that can be opened with a spreadsheet tool such as Excel.

To save the data:

1. Right-click on any module row and choose either:
   - **Export Resource to ASCII file**
   - **Export Resource to CSV file**
2. In the dialog box, browse to where you want to save the data.
3. Enter the file name.
4. Click **Save**.

See Also "Commands in the Hierarchy View" on page 55

About Netlist Analyzer

Netlist Analyzer works with Lattice Synthesis Engine (LSE) to produce schematic views of your design while it is being implemented. (Synplify Pro also provides schematic views.) Use the schematic views to better understand the hierarchy of the design and how the design is being implemented.

To start Netlist Analyzer:

1. Synthesize the design with LSE.
2. Choose **Tools > Netlist Analyzer**.

   The Netlist Analyzer window opens with the RTL netlist showing.
3. In the window's tool bar, click one of the following buttons:

   RTL View to see the design's logic (gates and registers) after LSE has optimized it

   Technology View to see the design after synthesis and translated into Lattice primitive modules

   Post-Mapping View (available only after mapping) to see how the design is mapped to the device's architecture

   Open External File to browse to a desired netlist database (.vdb) file

   A new tab opens and is filled as described below.

About Netlist Analyzer Views The Netlist Analyzer window has four parts:

- Tool bar provides buttons for various functions.
- Netlist browser provides nested lists of module instances, ports, nets, and clocks.
- Schematic view shows a schematic of the design. Depending on the size of the design, the schematic may be made of multiple sheets.
- Mini-map, which is a miniature view of the sheet, helps you pan and zoom in the schematic view.

Netlist Analyzer can have multiple schematics open. The open schematics are shown on tabs along the bottom of the window.

Bold lines are buses. Green lines are clock signals.

There are several ways to adjust the view of a schematic and to navigate through the hierarchy.

Setting Netlist Analyzer Options

With Netlist Analyzer you can specify labels in the schematics, how the hierarchy gets expanded, the highlight color, and the size of the sheets.

To change tool options:

1. Choose **Tools > Options**.

   The Options dialog box opens.
2. In the left-hand list, find and expand **Netlist Analyzer**.
3. See the following instructions for specific options.
4. When finished, click **OK**.

To specify labels:

1. Under Netlist Analyzer, select **Text**.
2. Select the labels that you want to see in the schematics.

To create a schematic bookmark:

1. Under Netlist Analyzer, select **Bookmarks**
2. In the Add Bookmark dialog box, add a name for the bookmark. All bookmarks are associated with a specific .vdb file with a time stamp. This feature allows you to save a schematic graph after a series of operations, such as filter and expand, for future reference. This features also allows you to save and restore different critical paths of the design.

To control how the hierarchy gets expanded:

1. Under Netlist Analyzer, select **Schematic**.

2. Adjust the values of the following items:
   - Hierarchy Depth: The number of levels to show when using the Dissolve Instances command. The default, 0, means to show all levels.
   - Levels of Logic to expand: The number of additional levels to show with the Expand To/From command.
   - Search depth to expand: The number of additional levels to show with the Expand to FF/IO and Expand to Primitive commands. The default, 0, means to show all levels.

To change the colors:

1. Under Netlist Analyzer, select **Schematic**.
2. Click on the **Select Color** or **Highlight Color** block. The select color is used to outline objects that you select in the schematic view. The highlight color is used to outline objects that you want to keep highlighted after selecting them.

   The Select Color dialog box opens.
3. Select a color by doing one of the following:
   - Click one of the basic colors on the left.
   - Click in the color pallet on the right.
   - Enter values in the boxes at the bottom right. Work in one column or the other as they are two separate methods of coding color.

   The rectangle in the lower-middle changes to show the selected color. You can change the darkness with the vertical slider at the upper-left.
4. Click **OK**.

   In the Options dialog box, the color block changes to match.
5. If you want to see the effect of your change in the views, drag the Options dialog box aside and click Apply.

To change the sheet size:

1. Under Netlist Analyzer, select **Sheet**.
2. Change the width and height as desired.

Adjusting the Schematic View

Netlist Analyzer schematics are usually displayed on many sheets, depending on the size of the design. You may also have a few different schematic views open at the same time.

There are a few ways to change the sheet being displayed and the arrangement of the views. If you are dissatisfied by the layout of a sheet, you can try regenerating it.

To change the sheet being displayed:

- Do one of the following:
  - Click the Next Sheet button.
  - Click the Previous Sheet button.
  - Click the arrow in the Goto Sheet button. In the drop-down menu, choose a sheet number.
  - Click the name at the end of a net on the left or right side of a schematic (but not a port). This takes you to the continuation of the net on another sheet.

To arrange views within the Netlist Analyzer window:

Anytime you open a schematic view it, along with its netlist browser, fills the Netlist Analyzer window. There is a tab for each view along the bottom of the window.

- To change the arrangement, do one of the following:
  - Click the Tile button to change the views into smaller windows that are arranged so that you can see them all at the same time.
  - Click the Cascade button to change the views into smaller windows that are arranged in a cascading, overlapping fashion.

The tabs remain in all modes. Click a tab to bring its window to the top.

In the tile and cascade modes, the schematic windows can be resized and dragged within the Netlist Analyzer window. Maximizing a window restores the tab mode.

To change the layout in a sheet:

- Right-click in the schematic and choose **Regenerate**.

  The schematic might change to a clearer layout. However, there will often be no change as the layout is already the best that Netlist Analyzer can do.

Zooming and Panning

You can zoom and pan within the Netlist Analyzer schematics using a variety of methods. These include toolbar commands, dragging with the schematic, and Netlist Analyzer's mini-map.

Toolbar Commands The following commands are available on the Diamond toolbar.

Zoom In – enlarges the view of the entire layout.

Zoom Out – reduces the view of the entire layout.

Zoom Fit – reduces or enlarges the entire layout so that it fits inside the window.

Zoom To – enlarges the size of one or more selected objects on the layout and fills the window with the selection.

Pan – enables you to use the mouse pointer to drag the layout in any direction. This command is useful for viewing a hidden area on the layout after zooming in.

Zooming with Function Key Shortcuts The following key combinations enable you to instantly zoom in or out from your keyboard.

- Zoom In – Ctrl++
- Zoom Out – Ctrl+-

Zooming with the Mouse Wheel The mouse wheel gives you finer zoom control, enabling you to zoom in or out in small increments.

- While pressing the **Ctrl** key, move the mouse wheel forward to zoom in and backward to zoom out.

Zooming by Dragging You can zoom by holding the right mouse button and dragging:

- To zoom to fit the window, drag up and to the left. The image adjusts to fill the window.
- To zoom out, drag up and to the right. The distance you drag determines the amount of zoom. 1 means half as big, 2 means ¼ as big, and so on. The image is reduced and centered in the window.
- To zoom in, drag down and to the right. The distance you drag determines the amount of zoom. 1 means twice as big, 2 means four times as big, and so on. The image is enlarged and centered in the window.
- To zoom in on a specific area, start at the upper-right corner of the area that you want to see better and drag to the lower-left corner of the area. The area that you drag across is adjusted to fill the window.

Zooming and Panning with Mini-Map The mini-map is a small display of a schematic at the lower-left of the Netlist Analyzer window. The mini-map shows the entire sheet with a purple rectangle marking the area that is actually showing in the schematic view.

If the mini-map is not showing, click the Mini-map button at the lower-right corner of the schematic view.

To enlarge the mini-map, click on one of the control tabs on the edge of the mini-map and drag.

To zoom, click on one of the control tabs of the purple rectangle and drag. As the rectangle changes size, the schematic view zooms in or out.

To pan, click in the purple rectangle and drag it so that it covers the area that you want to see. As you drag, the schematic view shifts.

Or, you can click on the spot that you want to see. The schematic view will jump to be centered on that spot.

Saving Schematics

You can save a schematic as either a PDF or SVG file. As a PDF file, the whole schematic is a single file with each sheet an 8.5-inch by 11-inch page. As an SVG file, only the currently displayed sheet is saved.

To save a schematic:

1. If saving as an SVG file, select the desired sheet.
2. Choose **File >** **Export**.

   The Export Schematic dialog box opens.
3. Browse to where you want to save the file.
4. Enter a file name.
5. Click the **Save as type** drop-down menu and choose the format.
6. Click **Save**.

Printing Schematics

Use the print functions of Diamond to print the schematics.

To print a schematic:

1. Choose **File >** **Print Preview**.

   The Print Preview window opens.
2. Expand the Print Preview window until you are comfortable with the display of the page.
3. To maximize the printout, click for landscape mode.
4. Click the Page setup button and adjust the paper size and margins if necessary.
5. Click the Print button.
6. Adjust the printer settings if necessary and click **Print**.

Navigating the Design with Netlist Analyzer

There are several ways to explore a design. Most start with selecting one or more objects. Then right-click in the schematic view and choose a command from the drop-down menu.

These commands help you to:

- Focus on one part of the design
- Flatten the layers of hierarchy
- Trace the paths between objects
- Expand selected parts of the schematic

Most commands can be reversed by clicking the Backward button. They can be repeated by clicking the Forward button. To jump back to the original view after several commands, right-click in the schematic view and choose **Restore Current Schematic**.

While exploring a design, you may see a need to set a synthesis constraint. If so, you can get a quick start by dragging the port, net, or register to the relevant tab of LDC Editor. See "Applying Lattice Synthesis Engine Constraints" on page 366.

To select objects:

- Click the Select button and do one of the following:
  - Click on the object in the schematic view. Ctrl-click to select more objects.
  - Click on the object in the netlist browser. Ctrl-click to select more objects.
  - Click and drag to draw a selection rectangle around one or more modules in the schematic view. This method only selects modules, not ports or nets.
  - Right-click in the schematic view and choose **Select All Schematic > Instances**. This command selects all module instances showing in all sheets of the schematic.
  - Right-click in the schematic view and choose **Select All Schematic > Ports**. This command selects all ports showing in all sheets of the schematic.
  - Right-click in the schematic view and choose **Select All Sheet > Instances**. This command selects all module instances showing in the current sheet of the schematic.
  - Right-click in the schematic view and choose **Select All Sheet > Ports**. This command selects all ports showing in the current sheet of the schematic.

You can keep an object highlighted, but not actively selected, by right-clicking in the schematic view and choosing **Highlight**. This outlines the selected objects in the highlight color, making them easy to spot, while you continue to explore the design. Highlighted objects are not "selected" for commands. Highlighting can be erased if you use the Backward button enough to hide the highlighted object.

To erase highlighting and return an object to normal display, select the object. Then right-click and choose **Unhighlight**.

Focusing on Part of the Design

Often you want to focus on just one part of the design. You can filter the schematic, reducing it to selected modules, trim away objects that are in the way, or ask for more details of a single module.

To filter the schematic:

1. In either the netlist browser or schematic view, select one or more modules of interest.
2. Click the Filter Schematic button.

   The schematic is replaced with just the selected modules. Only nets connecting the selected modules are included. Use other commands to expand the schematic.

To filter with signal paths:

1. Select one or more modules of interest.
2. Right-click anywhere in the schematic view and choose **Isolate Path**.

   The schematic is replaced with just the selected modules *and* modules connected to the selected modules. Only nets connecting the modules are included.

To trim the schematic:

1. Select one or more objects that you do *not* want to see.
2. Right-click anywhere in the schematic view and choose **Less**.

   The selected objects disappear. If the objects include a branch of a net, the remaining branches become dotted lines to show that part of the net is hidden.

To see the structure inside a module:

1. Click the Push down/Pop up button.
2. Drag the cursor to the module.

   If there is further hierarchy to see (that is, the module is not already a primitive), the cursor changes to a blue down-pointing arrow.
3. Click in the module with the down-pointing arrow.

   The schematic changes to show just the structure of the module at the next lower level of hierarchy. To go further down the hierarchy, click in one of these modules with the down-pointing arrow.

After you've started pushing down, the cursor shows a green up-pointing arrow when it is not over a module. To go back up the hierarchy, click when this up-pointing arrow is showing.

Remember that if you want to perform other commands in the schematic, you first need to click the Select button.

To quickly push down or pop up:

There is an easy way to push down into a module while staying in select mode:

- Right-click the module and drag straight down until the words “push down” appear. Then release the mouse button.
- To go back up the hierarchy, right-click anywhere in the schematic view and drag straight up until the words “pop up” appear. Then release the mouse button.

To work on just one module:

1. Go to the Hierarchy view. If it is not showing, choose **View > Show Views > Hierarchy**.
2. Right-click on the desired module and choose **Goto RTL Definition**.

   Netlist Analyzer opens with the RTL view of the module. The rest of the design is not available in this schematic.

Flattening the Design

Going back and forth through multiple layers of hierarchy can be confusing. An alternative is to flatten the design, displaying all the layers together. This makes for a much bigger, much more detailed schematic. But you do not have to keep track of where you are in the hierarchy. You can flatten the entire schematic or just selected modules.

To flatten the entire schematic:

- Right-click anywhere in the schematic view and choose **Flatten Schematic**.

  The entire design is shown in terms of primitives.

To return to the original schematic, right-click and choose **Unflatten Schematic**.

To flatten a module:

1. Select the module.
2. Right-click and choose **Dissolve Instances**.

   The schematic expands to include the selected module’s primitives. Boundaries, ports, and labels of the module and any submodules are included. So you can still identify and select the modules and their ports.

Tracing Paths between Objects

In a complicated schematic it can be difficult to trace a signal but Netlist Analyzer can do it for you.

To trace a signal path:

1. Select two or more objects. They can be on different sheets.
2. Right-click and choose **Expand > Expand Paths**.

   Nets and modules between the selected objects are highlighted. Modules along the paths are expanded to show primitives.

Expanding from a Module

After focusing a schematic to one or a few modules, you may want to see what the modules are connected to.

To see all the nets attached to a module:

1. Select the module.
2. Right-click and choose one of the following:
   - **Show Connectivity**

     Highlights all nets attached to the selected module, expanding the schematic as necessary.
   - **Expand > Expand To/From**

     Highlights all nets attached to the selected module and all primitives attached to those nets, expanding the schematic as necessary.

To see the next higher level of the hierarchy:

1. Select a module, right-click and choose **Show Context**.
2. If nothing happens, click the Push down/Pop up button. Move the cursor to the schematic view. If you see the green up-pointing arrow, click it.

Depending on the situation, one the above shows the schematic of the next higher level of the hierarchy.

Expanding from a Port

Instead of expanding all the nets from a module, you can expand the net from a single port. If the module is not a primitive, you have the choice of expanding outward or inward to a lower level of the module's hierarchy. Make this choice by selecting the port pointing out or the port pointing in.

To see what a single port is attached to:

1. Double-click the port.

   The schematic expands to show the port's net and one more object that is on the net.
2. If the net shows as a dashed line, you are seeing only part of the net. Double-click the port again to see another branch of the net.

If the net shows as solid line, you are seeing the full net.

If the full net is already visible, nothing happens.

To see a fuller signal path from a single port:

1. Select the port. If the port is on a module, take care whether it is pointing out or in.
2. Right-click and choose **Expand.**

   A sub-menu drops down.
3. Choose one of the following from the sub-menu:

   - **Expand to FF/IO**

     Highlights the net until it finds a flip-flop or an I/O pin, expanding the schematic as necessary. Modules along the path are expanded to show primitives.
   - **Current Level > Expand to FF/IO**

     Same as above except that modules are *not* expanded.
   - **Expand to Primitive**

     Highlights the net and all attached primitives, expanding the schematic as necessary.
   - **Current Level > Expand to Primitive**

     Same as above except that modules are *not* expanded.

Expanding from a Net

You can expand a net to show all connected primitives and ports or to show just the primitive or port driving it.

To see all modules on a net:

1. Select the net.
2. Right-click and choose **Go to Connected Instances**.

   Highlights the selected net and all connected primitives and ports, expanding the schematic as necessary.

To see just the driver of a net:

1. Select the net.
2. Right-click and choose **Go to Driver**.

   Highlights the selected net and the driving primitive or port, expanding the schematic as necessary.

Searching in Netlist Analyzer

With large designs, finding objects may be easier with a text search. The Find function in Netlist Analyzer allows you to search for objects in the schematic using instance, symbol, port, and net names. Found objects can be selected in the netlist browser and schematic view.

To search the design in Netlist Analyzer:

1. Choose **Edit >** **Find**.

   The Find dialog box opens.

2. Select the type of object that you want to find by clicking one of the tabs at the top of the dialog box.

3. Select the part of the design that want to search by selecting a level in the Search box.

4. At the bottom-left of the dialog box is the Un-Highlight Search box. Click in this box and type in your search term. You can use the following wildcards:

   - ***** for any number of characters
   - **?** for a single character

   Or, click the arrow at the end of the box and choose from previously used search terms.

5. Click either **Find 200** or **Find All**.

   The found objects are listed in the UnHighlight box.

6. If you clicked Find 200 and want to see more, click either **Next 200** or **Find All** until satisfied.

7. To highlight found objects in the schematic view, do one of the following:

   - For just some of the objects, select the desired objects in the UnHighlight box and then click the -> (right arrow) button.
   - For all of the found objects, click the **All** -> button.

   The selected objects move to the Highlight box and are highlighted in the netlist browser and schematic view. But the schematic view does not expand to show objects in other layers or on other pages.

8. To reduce the number of objects in the Highlight box, select unwanted objects and then click the <- (left arrow) button. You can find objects in the Highlight list using the Highlight Search box. Anything entered in the Highlight Search box is automatically searched for and selected.

9. When you have the Highlight list the way you want, click **Close**.

   Highlighted objects in the schematic view stay highlighted after the Find dialog box closes.

Getting More Information about an Object

You can get more information about an object in the schematic view such as names, number of fanouts, and the pins that nets connect to. You can also get the source code for the object.

To get more information about an object:

1. Select one or more objects.
2. In the schematic view, right-click the object of interest and choose **Properties**.

   The Properties dialog box opens showing a list of various properties of the object.

   If the object is a module, the dialog box also has a drop-down menu listing all the ports of the module. Choosing a port shows the properties for that port.

   If the object is a bus, the dialog box has a drop-down menu listing all the nets that make up the bus. Choosing a net shows the properties for that net.
3. When finished, click **Close**.

To get just the hierarchy name of an object:

1. Select one or more objects.
2. In the schematic view, right-click and choose **Copy**.
3. Paste into any text editor.

   The format of the name will be similar to this example:

   ```
   {i:UART_INST/u_txmitt/add_24}
   ```

   The first letter is i for instance, n for net, or p for port.

To get the source code for an object:

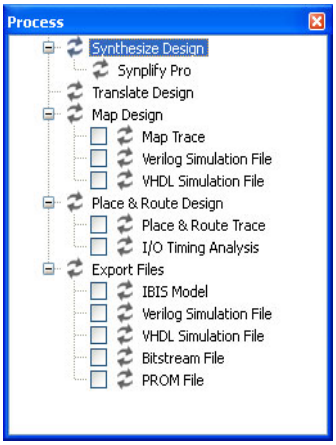
1. Select one or more objects.
2. In the schematic view, right-click the object of interest and choose **Jump to > Jump to HDL File**.

   Source Editor opens with the code highlighted.

Running Processes

A process is a specific task in the overall processing of a source or project. Typical processing tasks include synthesizing, mapping, placing, and routing. You can view the available processes for a design in the Process view. Some of the processes are device-dependent, such as JEDEC File and Bitstream File.

Process View



Processes are grouped into categories according to their functions.

- **Synthesize Design**

  By default, this process runs the Synplify Pro Synthesis tool in batch mode to synthesize the associated HDL design. For details, see "Synthesizing the Design" on page 578.

  If you are using Lattice Synthesis Engine (LSE) as the synthesis tool (**Project > Synthesis Tool**, see "Selecting a Synthesis Tool" on page 89 for details), This process will be changed to **Lattice Synthesis Engine**. Click on this process and Lattice Synthesis Engine (LSE) runs in batch mode to synthesize the design.

- **Translate Design**

  This process allows you to convert the EDIF file from the synthesis program to a Lattice internal database. If the design utilizes Lattice modules (such as Dual-Port RAM, Multiplier, ROM, Shift-Registers, and Adder/Subtractor), the modules will be expanded in this process. For details, see "Translating the Design Database" on page 588.

  Note for LSE

  If you have chosen LSE as the synthesis tool, the Translate Design process cannot be seen in the Process view. The translate design process is included in the Synthesize Design process.

- **Map Design**

  This process maps a design to an FPGA. Map Design is the process of converting a design represented as a network of device-independent components (such as gates and flip-flops) into a network of device-specific components (for example, configurable logic blocks). For details, see the Implementing the Design > Mapping section in the online Help.

  - **Map Trace**

    Trace can be used to run timing analysis in the post-mapping stage in the FPGA design flow. You can run this process after having mapped a design. Trace will create a timing report file (.twr) that will help a designer to determine where timing constraints will not be met. In

post-map timing analysis, Trace will determine component delay values and substitute zero values for missing routing delays. If path delay constraints that are unmet at this stage will be included in the report and may have to be relaxed.

- **Verilog Simulation File**

  This process backannotates the mapped design without timing information so that you may run a simulation of your design. The backannotated design is a Verilog netlist.

- **VHDL Simulation File**

  This process backannotates the mapped design without timing information so that you may run a simulation of your design. The backannotated design is a VHDL netlist.

- **Place & Route Design**

  After a design has undergone the necessary translation to bring it into the Native Circuit Description (.ncd) format, you can run the Place & Route Design process. This process takes a mapped physical design .ncd file, and places and routes the design, and outputs a file that can then be processed by the design implementation tools. For details, see "Place and Route" on page 602.

  - **Place & Route Trace**

    Trace can be used to run timing analysis in the post-routing stage in the FPGA design flow. You can run this process after having routed a design. Trace will create a timing report (.twr) that will allow a designer to verify timing. In post-route timing analysis, Trace analyzes your path delays and will report where these occur in the design.

  - **I/O Timing Analysis**

    This process runs I/O timing analysis and generates an I/O Timing Report. For each input data port in the design, this process generates the setup and hold time requirements and min/max clock-to-out delay for every output port. The computation is performed over all performance grades available for the device and at the voltage and temperature specified in the preference file. I/O timing analysis also automatically determines the clocks and their associated data ports.

  - **Thermal Analysis**

    (MachXO3L/LF 9400C/E devices only). This process must be run for MachXO3L/LF 9400C/E devices before running Bitstream File and JEDEC File processes. The Thermal Analysis process calculates the maximum safe ambient temperature for the user design implemented in MachXO3-9400C and mounted on three standard boards: JEDEC, Small, and Medium boards. This is to assist designers to determine if their design, implemented in MachXO3L/LF 9400C/E device, could safely be used in their application environment.

    The thermal analysis software uses the clock specified in the TWR file. If the clock frequency is not set in the TWR file, the thermal analysis process errors out. The designer can either set the clock frequency in the strategy section or in the timing constraint file which will be reported in the TWR file after running Timing Analysis.

The Thermal Analysis report will include Device Operation Conditions, Operating Clock frequency, Percentage of LUT utilization; Percentage of I/O Utilization, and Activity Factor information.

The report contains Thermal Impedance and Maximum Ambient temperature based on three different Board Mounts, No Heat Sink, and No Air Flow.

By default, the Thermal Analysis process is checked. Uncheck this box if you do not wish to run this process.

- **Export Files**

  You can check the desired file you want to export and run this process.

  - **IBIS Model**

    This process generates a design-specific IBIS model file (<*project_name*>.ibs).

    IBIS stands for I/O Buffer Information Specification. IBIS models provide a standardized way of representing the electrical characteristics of a digital IC's pins (input, output, and I/O buffers). For details, refer to "Lattice Semiconductor IBIS Models" on page 798.

  - **Verilog Simulation File**

    This process backannotates the routed design with timing information so that you may run a simulation of your design. The backannotated design is a Verilog netlist.

  - **VHDL Simulation File**

    This process backannotates the routed design with timing information so that you may run a simulation of your design. The backannotated design is a VHDL netlist.

  - **JEDEC File**

    This process produces a JEDEC file for programming the device. The JEDEC Standard is the industry standard for PLD formats. In the Diamond software, JEDEC refers to the fuse map of your design for the selected device. For details, refer to "Bit Generation" on page 639.

  - **Bitstream File**

    This process takes a fully routed physical design or Native Circuit Description (.ncd) file as input and produces a configuration bitstream (bit images). The bitstream file contains all of the configuration information from the physical design defining the internal logic and interconnections of the FPGA, as well as device-specific information from other files associated with the target device. For details, refer to "Bit Generation" on page 639.

  - **PROM File**

    This process takes a bitstream (.bit) file as input and generates output files in one of several PROM file formats. The output files are used to program PROMs. For details, refer to "PROM Generation" on page 648.

See Also "Managing Project Sources" on page 34

- "Process State" on page 74
- "Starting a Process" on page 75
- "Forcing a Process to Run" on page 75
- "Stopping a Process" on page 76
- "Refreshing Process State" on page 76
- "Cleaning Up Processes" on page 76

Process State

The Process view always shows the state of the processes in the active implementation. Each state is identified with different icons as shown in the table below.

Process State	Icon	Description
Initial		The process has never been run or an input dependency time stamp has changed since it was last run.
Completed		The process has been run successfully.
Warning		The process or report has been run successfully, but warning messages were issued.
Error		The process did not complete successfully.

Conditions that Re-initialize Process State A process or report is re-initialized under the following conditions:

- A process state earlier in the sequence has changed.
- An input file of the process or report has changed since the last run. For example, changing the .lpf file will cause the map process to be rerun.

  Note

  Input file changes are only detected if touched by the Lattice Diamond Source Editor or a process or report triggered by the Lattice Diamond main window. Changes applied by third-party text editors will NOT be detected.

- Process settings have changed.
- Target device has changed. All processes listed in the Process view in the sequence later than Translate Design will be re-initialized.
- The Process > Rerun All function is run.

See Also "Running Processes" on page 70

- "Starting a Process" on page 75
- "Forcing a Process to Run" on page 75

- "Stopping a Process" on page 76
- "Refreshing Process State" on page 76
- "Cleaning Up Processes" on page 76

Starting a Process

You can start a process on a single source file or on the entire project.

To start a process, do one of the following:

- Select the process in the Process view. Choose **Process >**  **Run**.
- Right-click the process and choose **Run**.
- Double-click the process.

See Also

- "Running Processes" on page 70
- "Process State" on page 74
- "Forcing a Process to Run" on page 75
- "Stopping a Process" on page 76
- "Refreshing Process State" on page 76
- "Cleaning Up Processes" on page 76

Forcing a Process to Run

If the process is up-to-date (indicated by a check mark to the left of the process), it will not run again. However, you can force a process to run by doing the following:

- Choose **Process >**  **Rerun** to start the selected process and run all the intermediate steps, even if the process is up-to-date. Using Rerun will force the selected tool to do a clean run overwriting any previous results.
- Right-click the highlighted process and choose **Rerun**.

You can use **Process >**  **Rerun All** to force all the processes before the one you selected to rerun. Using Rerun All will also force all tools to do a clean run overwriting any previous results.

See Also

- "Running Processes" on page 70
- "Process State" on page 74
- "Starting a Process" on page 75
- "Stopping a Process" on page 76
- "Refreshing Process State" on page 76
- "Cleaning Up Processes" on page 76

Stopping a Process

If you want to stop running certain process, choose **Process >**  **Stop**.

See Also ▶"Running Processes" on page 70

- "Process State" on page 74
- "Starting a Process" on page 75
- "Forcing a Process to Run" on page 75
- "Refreshing Process State" on page 76
- "Cleaning Up Processes" on page 76

Refreshing Process State

You can choose **Design > Refresh Design** to refresh the process state. For example, after running a certain process, you have made changes to a source file using a source editor outside Lattice Diamond. Then you can use Refresh Process to check the most recent state of the processes. This will also update the Hierarchy view and Reveal Inserter with any design changes.

See Also ▶"Running Processes" on page 70

- "Process State" on page 74
- "Starting a Process" on page 75
- "Forcing a Process to Run" on page 75
- "Stopping a Process" on page 76
- "Cleaning Up Processes" on page 76

Cleaning Up Processes

You can use the **Process > Clean Up Process** command to reset all the processes as shown in the Process view. This command returns the status of all of the processes back to their initial  state.

Setting Security Options

Some FPGA device families contain a hard-wired Advanced Encryption Standard (AES) engine that enables you to download an encrypted bitstream into the device. The ECP5, LatticeECP2S, LatticeECP2MS, LatticeECP3, and LatticeXP2 devices are equipped with this AES technology for downloading encrypted bitstream. MachXO3D devices have AES technology and ECDSA Authentication capability.

The Lattice Diamond software provides the Security Setting Tool that enables you to set up both the configuration security and an encryption key for those devices.

IMPORTANT!

To turn on this security feature, you need to install the Control_Pack_Encryption_Security.

Unencrypted JEDEC (.jed) and Bitstreams (.bit) can also be encrypted using Deployment Tool. Refer to:

- "Encrypting a JEDEC with Deployment Tool" on page 1000
- "Encrypting a Bitstream with Deployment Tool" on page 1001

See Also "Security Setting Dialog Boxes and Security Files" on page 77

- "Setting Security and Encryption for FPGA Devices with Security Setting Tool" on page 78
- "Setting Security and Encryption for MachXO3D Devices with Security Setting Tool" on page 80
- TN1212 - LatticeXP2 Advanced Security Programming Usage Guide
- TN1215 - Advanced Security Encryption Key Programming Guide for ECP5, LatticeECP3, and LatticeECP2/MS Devices

Security Setting Dialog Boxes and Security Files

The security setting tool consists of several successive dialog boxes, with which you can specify security settings for your target device and set a password to encrypt your security settings.

The Security Settings dialog box enables you to specify a flash protect key for LatticeXP2, and an encryption key for all other devices.

Once you have specified all the security settings, an encrypted .bek file will be generated in the project folder to hold the security settings. The .bek file contains readable header information that identifies the project, software version, device used, and creation date. The header is not encrypted so that you can identify the encrypted file.

If you have specified the flash protect key for LatticeXP2, a .key file will also be created in the project folder. The .key file holds the flash protection key.

The Enter Password dialog box and Change Password dialog box allow you to set a password to further encrypt your security settings. This password will

be used by Lattice Diamond to encrypt the .bek file and Programmer to decrypt the .bek file.

IMPORTANT!

You must track and remember all the keys and passwords you have specified. They will not be stored in any of the design files.

See Also ▶"Setting Security Options" on page 76

- "Setting Security and Encryption for FPGA Devices with Security Setting Tool" on page 78
- "Encrypting a JEDEC with Deployment Tool" on page 1000
- "Encrypting a Bitstream with Deployment Tool" on page 1001
- TN1212 - LatticeXP2 Advanced Security Programming Usage Guide
- TN1215 - Advanced Security Encryption Key Programming Guide for ECP5, LatticeECP3, and LatticeECP2/MS Devices

Setting Security and Encryption for FPGA Devices with Security Setting Tool

To run the Security Setting tool and set security settings:

1. In Lattice Diamond, choose **Tools > Security Setting**.

   The Enter Password dialog box opens. Because the project does not yet contain a .bek file, the dialog box shows LATTICESEMI as the default password.

2. To change the password, click **Change** to open the Change Password dialog box.
3. In the Enter Password field, type in a password of at least eight but no more than sixteen characters.
4. Type the password again in the Verify Password field and click **OK**.

   The Security Settings dialog box opens.

5. To set up keys, check the **Advanced Security Settings** option in the Security Settings dialog box.

   For LatticeXP2 devices, you have the option to set Flash Protect Key only or both Encryption Key and Flash Protect Key.

   For all other devices, you can only set the Encryption Key.

6. Select a key format from the Key Format drop-down list, and then type a key in the text boxes as follows, depending on the format selected:

- For the default ASCII format, type alphanumeric values using up to 16 characters.
- For hex format, type values of 0 through F, using up to 32 characters.
- For binary format, type 0 and 1 values, using up to 128 characters.

7. Click **OK** to apply the settings.

   A Security Setting message informs you that the encrypted files have been produced.

8. Click **OK** to close the Security Setting dialog box.

Note

The .bek and .key files are stored in the root project directory, instead of the implementation directory. In case you have more than one implementation, the .bek and .key files are generated for the active implementation.

See Also "Setting Security Options" on page 76

- "Security Setting Dialog Boxes and Security Files" on page 77
- "Encrypting a JEDEC with Deployment Tool" on page 1000
- "Encrypting a Bitstream with Deployment Tool" on page 1001
- TN1212 - LatticeXP2 Advanced Security Programming Usage Guide
- TN1215 - Advanced Security Encryption Key Programming Guide for ECP5, LatticeECP3, and LatticeECP2/MS Devices

Setting Security Password for MachXO2/MachXO3L/LF Devices with Security Setting Tool

To run the Security Setting tool and set security settings:

1. In Lattice Diamond, choose **Tools > Security Setting**.

   The Enter Password dialog box opens. Because the project does not yet contain a .key file, the dialog box shows LATTICESEMI as the default password.

2. To change the password, click **Change** to open the Change Password dialog box.
3. In the Enter Password field, type in a password of at least eight but no more than sixteen characters.
4. Type the password again in the Verify Password field and click **OK**.

   The Security Settings dialog box opens.

5. Click **OK** to apply the settings.

6. Click **OK** to close the Security Setting dialog box.

Note

The .key files are stored in the root project directory, instead of the implementation directory. In case you have more than one implementation, the .key files are generated for the active implementation.

See Also ▶"Setting Security Options" on page 76

- "Security Setting Dialog Boxes and Security Files" on page 77

Setting Security and Encryption for MachXO3D Devices with Security Setting Tool

The Security Setting Tool (Security GUI) allows you to generate and/or verify keys used for bitstream obfuscation. Additionally, ECDSA Authentication Signature provides the capability to generate and/or verify a signature for any data.

IMPORTANT

To turn on this security feature, you need to install the Control_Pack_Encryption_Security.

See Also ▶"Security Settings with Security Setting Tool" on page 80

- "Signature Authentication with Security Setting Tool" on page 85

Security Settings with Security Setting Tool

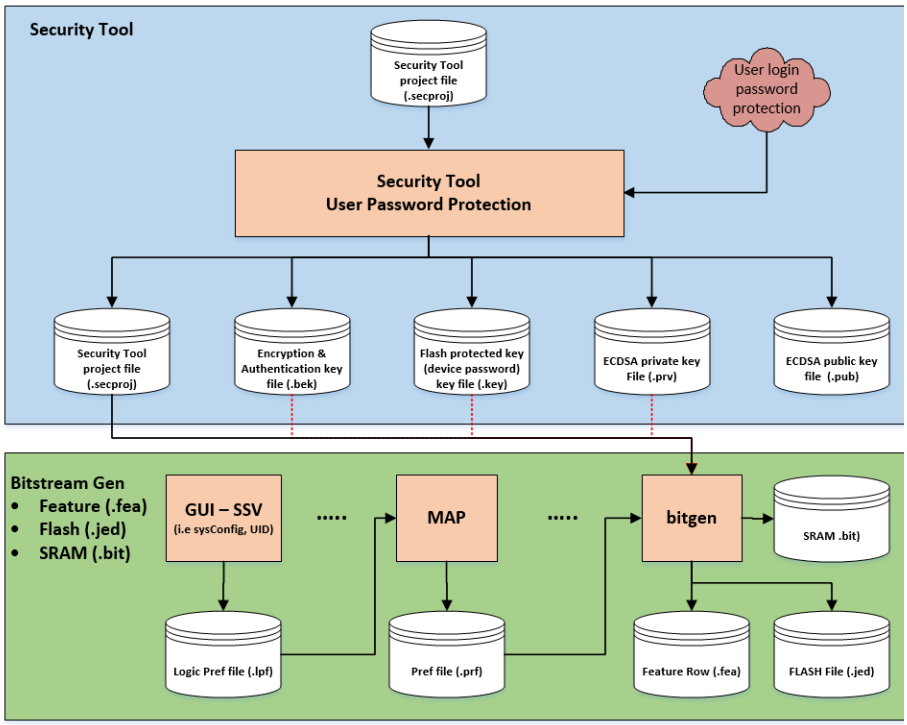
Security Settings provides the user entry for Flash protection (128-bit device password), AES-256 Encryption, and ECDSA Authentication.

User login password protection is required in Security GUI. You can either define a new password or use the preset default LATICESEMI password.

Security GUI generates various output key files that are user login password protected:

- Security Tool project file (.secproj), which is encrypted by user login password.
- Flash Protect Key file (.key) contains 128-bit device password.
- Encryption Key file (.bek) contains 256-bit AES key, which is encrypted by user login password.

- Authentication (ECDSA) Private Key file (.prv) contains either the 256-bit Private Key or 256-bit Signature.
- Authentication (ECDSA) 512-bit Public Key file (.pub)



The level of bitstream obfuscation depends on the options selected in Security GUI.

Flash Protection (Device Password) Flash protection generates a 128-bit device password key.

To run the Security Setting tool and set security settings:

1. In Lattice Diamond, choose **Tools > Security Setting**.

   The Enter Password dialog box opens. Because the project does not yet contain a .key file, the dialog box shows LATTICESEMI as the default password.

2. To change the password, click **Change Password** to open the Change Password dialog box.
3. In the Enter Password field, type in a password of at least eight but no more than sixteen characters.
4. Type the password again in the Verify Password field and click **OK**.

   The Security Settings dialog box opens.

5. To set up device password, check the **Flash Protection (Device Password)** option in the Security Settings dialog box.

6. Select a key format from the Key Format drop-down list, Load from file, or Default Key.
   - Select a key from the Key Format drop-down list, and then type a key in the text boxes as follows, depending on the format selected:
     - For the default ASCII format, type alphanumeric values using up to 16 characters.
     - For hex format, type values of 0 through F, using up to 32 characters.
     - For binary format, type 0 and 1 values, using up to 128 characters.
   - Select a key from Load from File.
     - The Load Key from File dialog box opens, select the key file and click Open.
   - Select a key by clicking on Default Key button.
     - The tool generates LATTICESEMICONDU key.
7. Click **OK** to apply the settings.
8. Click **OK** to close the Security Setting dialog box.

AES Encryption The AES-256 encryption can be used without any authentication. Since AES is a symmetric-key algorithm, the same key is used for both encrypting and decrypting the bitstream.

To run the Security Setting tool and set security settings:

1. In Lattice Diamond, choose **Tools > Security Setting**.

   The Enter Password dialog box opens. Because the project does not yet contain a .key file, the dialog box shows LATTICESEMI as the default password.
2. To change the password, click **Change Password** to open the Change Password dialog box.
3. In the Enter Password field, type in a password of at least eight but no more than sixteen characters.
4. Type the password again in the Verify Password field and click **OK**.

   The Security Settings dialog box opens.
5. To set up AES key, check the **AES Encryption** option in the Security Settings dialog box.
6. Select a key format from the Key Format drop-down list, Load from file, or Auto Generated Key.
   - Select a key from the Key Format drop-down list, and then type a key in the text boxes as follows, depending on the format selected:
     - For the default ASCII format, type alphanumeric values using up to 32 characters.
     - For hex format, type values of 0 through F, using up to 64 characters.

- For binary format, type 0 and 1 values, using up to 256 characters.

- Select a key from Load from File.
  - The Load Key from File dialog box opens, select the key file and click Open.
- Select a key by clicking on Auto Generated Key button.
  - The tool generates a random 256-bit AES key in hex format.

7. Click **OK** to apply the settings.

   A Security Setting message informs you that the encrypted files have been produced.

8. Click **OK** to close the Security Setting dialog box.

ECDSA Authentication The Elliptical Curve Digital Signature Algorithm (ECDSA) offers a variant of the Digital Signature Algorithm (DSA), which uses the elliptical curve cryptography. The Authentication Private key (or Signature) and Public key are used to encrypt/decrypt the bitstream with ECDSA. The Public key size is 512-bits, the Private key size is 256 bits.

The ECDSA Authentication key pair (Private key and Public key) are verified. Alternatively, the Signature with the Public key can be verified given that you provide SHA-256 Signature and Public key pair. The Signature size is 64 Bytes.

An optional feature allows to verify the SHA-256 Signature against the SHA-256 Digest.

IMPORTANT

The SHA-256 digest is used to verify Signature and Public key and is **not** saved to Security project file.

To run the Security Setting tool and set security settings:

1. In Lattice Diamond, choose **Tools > Security Setting**.

   The Enter Password dialog box opens. Because the project does not yet contain a .key file, the dialog box shows LATTICESEMI as the default password.

2. To change the password, click **Change Password** to open the Change Password dialog box.
3. In the Enter Password field, type in a password of at least eight but no more than sixteen characters.
4. Type the password again in the Verify Password field and click **OK**.

   The Security Settings dialog box opens.

5. To set up the key pair, check the **ECDSA Authentication** option in the Security Settings dialog box.

   a. Check the **Public key and private key** option. Public key and Private key dialog boxes are enabled. Select a key format from the Key Format drop-down list, Load from file, or Auto Generated Key Pair.

- Select keys from the Key Format drop-down list, and then type a key in the text boxes as follows, depending on the format selected:
  - For the default ASCII format, type alphanumeric values using up to 32 characters for Private Key and 64 characters for Public Key.
  - For hex format, type values of 0 through F, using up to 64 characters for Private Key and 128 characters for Public Key.
  - For binary format, type 0 and 1 values, using up to 256 characters for Private Key and 512 characters for Public Key.
- Select Public key and Private key from Load from File.
  - The Load Key from File dialog box opens, select the Public key file and click Open.
  - The Load Key from File dialog box opens, select the Private key file and click Open.
- Select Public key and Private key by clicking on Auto Generated Key Pair button.
  - The tool generates random 512-bit Public and 256-bit Private key pair in hex format.

b. Check the **Public key and SHA-256 signature** option. Public key and SHA-256 signature dialog boxes are enabled. Both, Public key and Signature, has size of 512 bits. Select a key format from the Key Format drop-down list or Load from file.

- Select key and signature from the Key Format drop-down list, and then type a key in the text boxes as follows, depending on the format selected:
  - For the default ASCII format, type alphanumeric values using up to 64 characters for Public Key and Signature.
  - For hex format, type values of 0 through F, using up to 128 characters for Public Key and Signature.
  - For binary format, type 0 and 1 values, using up to 512 characters for Public Key and Signature.
- Select Public key and signature from Load from File.
  - The Load Key from File dialog box opens, select the Public key file and click Open.
  - The Load Key from File dialog box opens, select the Private key file and click Open.
- (Optional) Type in the SHA-256 digest. The digest size is 256 bits. Check **Verify** to verify the SHA256 signature against the SHA-256 digest; otherwise, Public key and Signature will be saved, but **not** verified as a valid pair.

6. Click **OK** to apply the settings.

   A Security Setting message informs you that either the encrypted files have been produced or public key and private key don't match. If the key pair cannot be verified, the output encrypted files are not generated.

7. Click **OK** to close the Security Setting dialog box.

ECDSA Authentication with AES encryption A merge of AES encryption/decryption with ECDSA authentication allows for additional level of bitstream obfuscation. First, the private/public key pair is used to encrypt/decrypt the bitstream with ECDSA. Second, the bitstream is encrypted/decrypted with the AES-256 key.

To generate/verify appropriate keys, follow sections "AES Encryption" on page 82 and "ECDSA Authentication" on page 83.

Signature Authentication with Security Setting Tool

Signature Authentication allows you to generate or verify a signature for any data file. To generate signature, you must provide the public-private key pair along the data file the signature is intended for. To verify signature, it is compared against a known public key.

Signature Authentication generates these output files:

- Signature file (.sig) contains 256-bit Signature.
- Digest file (.digest) contains 256-bit Digest.

Signature Generation Signature Generation tool generates Signature and Digest for any data file. Once generated, Signature is visible in the GUI's Signature window. Generated files are saved in the same directory as the data file.'

To run the Signature Generation:

1. In Lattice Diamond, choose **Tools > Security Setting**.

   The Enter Password dialog box opens. Enter your password or use the default password LATTICESEMI. Click **OK**.

   The Security Setting dialog box opens.

2. In the **Signature Authentication** tab, select **Signature Generation**. Provide the Data File, Public, and Private Key.
3. Select both keys.
   - Select Public and Private Keys from Load from File.
     - The Load Key from File dialog box opens, select the key file and click **Open**.
   - Generate Public and Private Keys by clicking on Auto Generated Key Pair button.
     - The tool generates a random 256-bit Private Key and 512-bit Public Key in hex format.
   - If desired, change the key format from the Format drop-down list:
     - For the default ASCII format, type alphanumeric values using up to 32 characters for Private Key and 64 characters for Public Key.

- For hex format, type values of 0 through F, using up to 64 characters for Private Key and 128 characters for Public Key.
- For binary format, type 0 and 1 values, using up to 256 characters for Private Key and 512 characters for Public Key..

- Select Data File from Load from File.
  - The Select Data File to generate or verify signature dialog box opens, select any data file and click **Open**.

4. Click **Generate** to generate a signature for the provided data file.

Signature Authentication message window appears stating the Signature and Digest were successfully created and saved.

Signature Verification Signature Verification allows you to confirm the validity of the data file's signature. The tool compares provided Signature against the provided Public Key. If the verification fails, an error message occurs in the Signature Authentication message box window.

To run the Signature Verification:

1. In Lattice Diamond, choose **Tools > Security Setting**.

   The Enter Password dialog box opens. Enter your password or use the default password LATTICESEMI. Click **OK**.

   The Security Setting dialog box opens.

2. In the **Signature Authentication** tab, select **Signature Verification**. Select Public Key from Load from File.
   - The Load Key from File dialog box opens, select the key file and click **Open**.
   - If desired, change the key format from the Format drop-down list:
     - For the default ASCII format, type alphanumeric values using up to 64 characters.
     - For hex format, type values of 0 through F, using up to 128 characters.
     - For binary format, type 0 and 1 values, using up to 512 characters.
3. Select Data File from Load from File.
   - The Select Data File to generate or verify signature dialog box opens, select any data file and click **Open**.
4. Select Signature from Load from File.
   - The Load Signature from File dialog box opens, select .sig file and click **Open**.
5. Click **Verify**.

Signature Authentication message window appears with pass or fail message.

Setting Options for Synthesis and Simulation

Diamond is integrated with the Lattice Synthesis Engine (LSE), Synplify Pro synthesis tool, and the ModelSim simulation tool. Besides the OEM tools, you can also let Diamond use LSE, your full-featured versions of Synplify Pro or Precision as the synthesis tool, and full-featured versions of ModelSim or Active-HDL as the simulation tool.

This section covers steps on selecting synthesis and simulation tools, defining file order for synthesis and simulation, customizing synthesis tool processes, specifying VHDL library name, and specifying the search path for Verilog include files. Also, how to perform interactive synthesis is discussed in this section.

About Lattice Synthesis Engine

For most devices, you have the option of using Lattice Synthesis Engine (LSE) as your synthesis tool instead of Synplify Pro for Lattice or another third-party synthesis tool.

LSE is a synthesis tool custom-built for Lattice products and fully integrated with Diamond. Depending on the design, LSE may lead to a more compact or faster placement of the design than another synthesis tool would do. LSE can be run from the Diamond main window or through the command line, similar to the other integrated synthesis tools.

Also, LSE offers the following advantages:

- Enhanced RAM and ROM inference and mapping, including:
  - multiple reads
  - dual-port RAM in write-through, normal, and read-before-write modes mapped to EBR
  - clock enable and read enable packing
  - mapping for the minimal number of EBR blocks
  - EBR mapping for minimal power
  - better support for wide-mode mapping
- Comprehensive GSR inference for area and timing
- Simpler flow to get an .ngd file: no ngdbuild command or Translate Design process required
- Post-synthesis Verilog netlist suitable for simulation

When considering LSE, note the following limitations:

- Not available with LatticeEC, LatticeECP, LatticeSC/M, or LatticeXP.
- IP must be generated with a synthesis tool other than LSE. So the synthesis of IP won't be affected by a change to LSE.
- Design code may need to be modified. Modifications may involve some VHDL requirements and inferring RAM, ROM, and I/O.

- SystemVerilog features are not supported.
- Does not include interclock domain paths in timing analysis.

See Also

- “Selecting a Synthesis Tool” on page 89
- Lattice Synthesis Engine Tutorial

Switching to Lattice Synthesis Engine

When changing to Lattice Synthesis Engine (LSE) from another synthesis tool, there are a few things to do:

- Consider creating a new design implementation. You can experiment with different settings without losing your previous work. See “Working with Implementations” on page 46.
- Modify the Synopsys Design Constraint (.sdc) file into an .ldc or create a new .ldc file. See “Lattice Synthesis Engine (LSE) Constraints” on page 1339 and “Applying Lattice Synthesis Engine Constraints” on page 366.
- Adjust the LSE strategy settings. See “LSE Options” on page 1129 and “Optimizing LSE for Area and Speed” on page 581.
- Review the LSE coding tips and consider applying them to your design. See “Coding Tips for Lattice Synthesis Engine (LSE)” on page 173.
- Review the instructions for integrated synthesis. There are some differences with LSE. See “Integrated Synthesis” on page 583.

See Also

-

Bus Naming in LSE Output

In the output files from Lattice Synthesis Engine (LSE), bus names are converted to individual signal names (sometimes known as “bit blasting”). LSE adds “_c_” to the individual signal names. For example, the bus name sum[0:3] becomes:

sum_c_0
sum_c_1
sum_c_2
sum_c_3

See Also

- “Consistent Bus Name Conversion” on page 1138

Selecting a Synthesis Tool

Diamond supports Verilog HDL and VHDL designs using Lattice Synthesis Engine (LSE), Synplify Pro, or Precision RTL as the synthesis tool.

- Diamond is fully integrated with LSE and Synplify Pro. “Fully integrated” means that you can set options and run synthesis entirely from within Diamond.
- Diamond also supports Mentor Graphics Precision RTL Synthesis but does not include the tool itself. If you want to use Precision RTL Synthesis, you need to get the software on your own.
- If you prefer, you can use other synthesis tools by running them independently of Diamond.

You can specify one of them as the synthesis tool for the currently active implementation of your project. Different implementations can have different synthesis tools specified.

To select a synthesis tool:

1. Make sure you have successfully installed LSE, Synplify Pro, or Precision.
2. From the Diamond main window, choose **Project > Active Implementation > Select Synthesis Tool**.

   The Project Properties dialog box appears with the active implementation selected.
3. In the Project Properties dialog box, double-click the synthesis tool in the Value column.
4. From the drop-down menu, select a tool.
5. Click **OK**.

If you have chosen **Synplify Pro**, you can either use Synplify Pro for Lattice or your own full-featured Synplify Pro. By default, Diamond will use the Lattice version. If you want to use Synplify Pro for Lattice, you can use the main window to start running processes. If you want to use your own full-featured Synplify Pro, you need to specify the directory in the Options dialog:

1. Choose **Tools > Options**.

   The Options dialog box opens.
2. In the **Environment > Directories** tab, clear the **use OEM** option.
3. Specify the installation path for Synplify Pro in the Synplify Pro field.
4. Click **OK** to save your settings.

If you have chosen **Precision** as the synthesis tool, you need to specify the directory in the Options dialog:

1. Choose **Tools > Options**. The Options dialog box opens.
2. In the **Environment > Directories** tab, specify the installation path for Precision in the Precision field.
3. Click **OK** to save your settings.

See Also ▶"Customizing Synthesis" on page 93

▶

Specifying VHDL Library Name

The Lattice Diamond software supports VHDL design synthesis using LSE, Synplify Pro, or Precision Synthesis. You can specify the library name for synthesizing individual VHDL sources and modules. By default, all project-related design sources are compiled into the "work" library.

The VHDL library name can also be specified in the Synplify Pro or Precision Synthesis tool. See Synplify Pro or Precision Synthesis Help for details.

To specify the VHDL library target in the main window:

1. In the File List view, right-click the VHDL source for which you want to specify a library name and choose **Properties**.
2. In the Project Properties dialog box, type in the library name you want in the Value field and click **OK**.

   The LSE, Synplify Pro, or Precision Synthesis tool will then use the specified library name to synthesize the selected VHDL source file or module.

Specifying Search Path for Verilog Include Files

The Lattice Diamond software supports Verilog design synthesis using LSE, Synplify Pro, or Precision Synthesis. If a Verilog file is added to your design and additional files are referenced via the include directive, you can specify the search path in Lattice Diamond for searching include files.

Note

If the include file in your Verilog file changed after you opened the relevant project in Diamond, you need to use the **Rerun All** or **Rerun** command to force the process to rerun so that Diamond can reflect the changes of the include file.

You can also specify a Verilog include file search path directly in LSE, Synplify Pro, or Precision. See "Running SYNTHESIS from the Command Line" on page 2480, and the Synplify Pro or Precision Synthesis Help for details.

Synplify Pro and LSE Search Path If you select Synplify Pro or LSE as the synthesis tool, you can only specify the search path for the entire project. Synplify Pro or LSE searches for the include file in the following order, and stops at the first occurrence of the include file it finds.

1. The directory of the file that specifies the 'include' directive

2. The directories that are specified in the “Search Path” option for the entire project

Precision Synthesis Search Path If you use Precision, you can specify the search path for the project and for each Verilog source file. Precision searches for the include file in the following order, and stops at the first occurrence of the include file it finds.

1. The directory of the file that specifies the 'include' directive
2. The directories that are specified in the “Search Path” option for the Verilog source
3. The directories that are specified in the “Search Path” option for the entire project

Specifying Search Path in Lattice Diamond

To specify the search path in Lattice Diamond:

1. In the File List view, do either of the following:
   - Right-click the implementation name if you want to specify the search path for the entire project.
   - (*For Precision Synthesis only*) Right-click the Verilog source for which you want to specify the search path.
2. Choose **Properties** from the right-click menu.

   The Project Properties dialog box opens.
3. In the dialog box, select **Verilog Include Search Path**, and do either of the following:
   - If you want to add only one search path, click the Value field to enter a path.
   - If you want to add multiple paths, click the **...** button from the Value field to get the Verilog Include Search Path dialog box. In this dialog box, click the  icon to browse for a new path and click **OK**. The new path is displayed in the Verilog Include Search Path dialog box. Repeat to add more paths. You can use the  and  button to arrange the order of the paths, or use the  button to delete a path. Click **OK** when you finish.
4. Click **OK** to exit the Project Properties dialog box.

   The search paths you added are applied. The Synplify Pro or Precision Synthesis tool will use the specified search paths to search for the include files referenced in your design other than the directory of the file that specifies the include directive.

Specifying Verilog and VHDL Parameters

You can add Verilog compiler directives and parameters or VHDL generics for the whole design in the properties of each design implementation. You may find this method an easier way to experiment with these settings.

To add Verilog compiler directives and parameters or VHDL generics:

1. In the File List view, right-click the implementation's name and choose **Properties**.

   The Project Properties dialog box opens showing the implementation's properties.

2. In the HDL Parameters row, click under **Value**.
3. Type the statements as a single text line with different statements separated by semi-colons (;).
4. When done, click **OK**.

Selecting a Simulation Tool

Diamond supports functional and timing simulation for Lattice FPGA devices using ModelSim or Active-HDL. You can specify either ModelSim or Active-HDL as the simulation tool for your project in the Simulation Wizard (**Tools > Simulation Wizard** from main window). For more details on how to use Simulation Wizard, see "Simulating the Design" on page 310.

Specifying Input Files for Simulation

Your design probably includes one or more test bench files that you want to use during simulation but do not want synthesized. You can specify this in the properties for each input file. New files are automatically marked for both synthesis and simulation.

To change the synthesis/simulation property of a file:

1. In the File List view, right-click the filename.
2. In the drop-down menu, choose **Include for** and then the desired property:
   - Synthesis and Simulation
   - Synthesis
   - Simulation

You can also set the synthesis/simulation property in the Project Properties dialog box. You can open the dialog box by choosing **Project > Property Pages**.

Defining File Order for Synthesis and Simulation

The File List view lists design input files in a specific order. The synthesis or simulation tool uses that order to sequentially synthesize or simulate each input file. You can adjust the file order by the “drag and drop” operations. The file list can include a combination of VHDL, Verilog, and schematic files. If you have not manually set a top-level module (in the Project Properties dialog box of the main window), usually the last file on the file list is treated as the top-level module.

File order is especially important for VHDL files. Package files must be first on the list because they are compiled before they are used. If design blocks are spread over many files, you should specify the following file order: the file containing the entity must be first, followed by the architecture file, and finally the file with the configuration. Verilog file order is typically important when ‘define statements are dedicated to a single file. This file should be compiled prior to any files that refer to the variables.

To define the file order for logic synthesis and simulation:

- In the Input Files section of the File List view, change the file order as you like by dragging the file name and dropping it to the required location.

  The new file order takes effect immediately. The order will be saved when you save the design project.

Customizing Synthesis

Lattice Diamond supports Verilog and VHDL synthesizing using Lattice Synthesis Engine (LSE), Synplify Pro, and Precision.

You can adjust synthesis tool options and perform the logic synthesis process for your Verilog or VHDL design within Lattice Diamond.

To adjust synthesis tool options and perform the synthesis process:

1. Choose **Project > Active Strategy >** <*synthesis tool*> **Settings**.
2. In the Strategies dialog box, by default, the option settings of the selected process, LSE, Synplify Pro, or Precision, are listed. You can use the **Display catalog** to filter the options you need to customize.
3. Double-click the Value field of the option. You can specify the option using the drop-down menu of the Value field.
4. Click **OK** to accept the new synthesis tool options and exit the Strategies dialog box.
5. In the Process view, double-click the Synthesize Design process to execute the selected synthesis tool.

If you prefer leveraging all the interactive features available in the synthesis tool while maintaining HDL source in the main window, see "Synthesizing the Design" on page 578 for more details.

Working with Run Manager

Large designs can quickly get complicated, and the best approach is not always obvious or definite. There may be many ways to get optimal performance out of your design—the key is to find one option that works.

In Diamond, each project can contain multiple source files, strategies, and implementations. Implementations are comprised of selected source files and only one strategy. You can create multiple implementations to bind different strategies.

After creating implementations, use Run Manager to run multiple synthesis and place and route passes, compare the results of multiple implementations for further analysis to get best solutions. Run Manager helps you to manage running the design implementation process with multiple project implementations (versions) and to compare the results. You can monitor progress, view reports, and quickly identify the best implementation.

See Also "Using Strategies" on page 48

"Working with Implementations" on page 46

Setting Up Run Manager

Before using Run Manager, consider if you want to do multiple place-and-route runs on individual implementations to get different timing and area results.

If you have a lot of implementations and don't want to run them all, you may want to hide some of them in Run Manager.

Also, consider your system's ability to run multiple processes. If you have several implementations running simultaneously in Run Manager, you may see performance problems because of RAM resource conflicts.

Setting Up Multiple Runs on an Implementation You can get multiple place-and-route runs, using different seed values, on a single implementation. The different values can produce significantly different timing and area results.

To set up multiple place-and-route runs on an implementation:

1. In the File List view, right-click on the strategy of the implementation and choose **Edit**. Strategies are listed in the Strategies folder. If you are not sure which strategy is being used, you can check the Implementation column in Run Manager. It includes the strategy name for each implementation.

The Strategies dialog box opens.

2. In the Process box, on the left, choose **Place & Route Design**.
3. Find Placement Iterations and double-click the Value box. Enter the number of place-and-route runs you want to make. (For more on this option, see "Placement Iterations" on page 1148.)
4. Find Placement Save Best Run and double-click the Value box. Enter the number of place-and-route runs you want to see. Run Manager will show you this many of the best results. (For more on this option, see "Placement Save Best Run" on page 1149.)
5. Click **OK**.

Hiding Implementations Run Manager normally shows all the implementations of the project. But you can choose to hide any of the implementations from Run Manager except for the active implementation.

To change which implementations are showing:

1. Choose **Tools > Run Manager**.

   Run Manager opens with a table listing the implementations available. The row in bold font indicates the active implementation.
2. If you want to hide the active implementation, right-click an implementation that you will continue to show and choose **Set As Active**.

   This implementation becomes the active one for the project. The row changes to bold type and, in File List view, the implementation is expanded. The previously active implementation becomes inactive and can be hidden.
3. In Run Manager's tool bar, click the Show/Hide Implementation button.

   The Show/Hide Implementation dialog box opens.
4. In one of the columns, select one or more implementations that you want to move to the other column.
5. Click the arrow button pointing to the other column.
6. When you have the Hidden and Show lists as you want them, click **OK**.

Improving System Performance You can improve your system's performance while running Run Manager by limiting the number of processes sharing the memory. Allowing fewer processes means your computer spends less time swapping data in and out of the hard drive. But fewer processes than the number of available processors could mean processors sitting idle. You will need to find a balance based on your computer and the way you tend to use Run Manager.

To improve performance:

1. Choose **Tools > Options**.
2. In the Options dialog box, choose **Environment > General**.

3. Find the **Maximum number of implementation processes in run manager** option and change the number. The best value is the number of processors in your computer or fewer.
4. Find the **Maximum number of multi-par processes in run manager** option and change the number. The best value is the number of processors in your computer or fewer.
5. Click **OK**.

Running Run Manager

After setting up Run Manager (see "Setting Up Run Manager" on page 94), you are ready to run your implementations.

To process multiple implementations using Run Manager:

1. Choose **Tools > Run Manager**.

   Run Manager opens with a table listing the implementations available. The row in bold font indicates the active implementation.
2. Select the implementations that you want to run by setting the check boxes. Select them all by clicking the check box in the heading row.
3. In Run Manager's tool bar, click the Run button. You can force implementations that have already been run (the Status column says Completed) by clicking the Rerun button.

   While running, you can pause or stop the runs. Select a running item by clicking anywhere in its row. Then click either the Pause or Stop button. Run Manager may take a few moments to respond.

   To continue a paused run, select it and click the Run button.

   Note

   The Process menu commands also control Run Manager, but they only affect the active implementation. This is true whether you choose the command from the Process menu in the menu bar or the right-click menu in the Process view.
4. When the runs are complete, compare the statistics that appear. These statistics provide a quick comparison of the quality of results. See "Reading the Run Manager Results" on page 97.

   If you set up an implementation to do multiple place-and-route runs, click the plus sign to show the individual results. The top scoring result for each implementation is shown in italics and will be used in any further processing with that implementation.
5. If you want to save the results, go to Run Manager's tool bar and click the Export button. In the Export Run Manager File dialog box, browse to where you want to save the results and enter a file name. Then click **Save**.

All the data showing in Run Manager are saved in a comma-separated values (.csv) file that can be opened with a spreadsheet tool such as Excel.

6. For more details, go to the Reports view. The Reports view shows a tab with a set of reports for each implementation. To jump to a specific report, in Run Manager, right-click the row for the implementation and choose **Show Report >** <*report*>. See "Viewing Logs and Reports" on page 101.

   Also, examine the Output view, including checking for any warning or error messages.

7. After studying the results, you can select one of the implementations and one of its runs for further development. See "Selecting an Implementation and Run" on page 99.

Reading the Run Manager Results

Run Manager displays results with a table. Every implementation in the project gets a row. If an implementation did multiple place-and-route runs, each run has a row under its implementation. If you don't see the separate place-and-route runs, expand the hierarchy tree under the implementation row.

Bold text shows the active implementation. Italic text shows the active run out of multiple place-and-route runs. This is the run that will be used in further processing of that implementation.

There are several columns showing the results for the implementations and the place-and-route runs. The columns are described below. Compare the results across rows to see which run worked best.

For the active implementation, you can see the status of the different processes by looking at the Process view. To see the status of another implementation, right-click the implementation in Run Manager and choose **Set As Active**.

For more details, check the Reports view. The Reports view shows a tab with a set of reports for each implementation run. See "Viewing Logs and Reports" on page 101.

Changing the Display You can adjust the display by re-arranging the columns and sorting the runs. By default, many of the columns are hidden but you can choose which ones you want to see.

To change which columns are displayed:

- Right-click in any column heading and choose the column that you want to hide or show. Columns that are showing have a check mark. The Implementation<Strategy> column cannot be hidden.

To re-arrange columns:

- Click on the heading of the column that you want to move and drag it to where you want it.

To change the width of columns:

- Do one of the following:
  - Right-click in any column heading and choose Adjust Column Width to Contents.

    All columns are adjusted based on their contents.
  - Click on the right-hand border of a column and drag to make that column narrower or wider.

To sort the runs:

1. Click in the heading of the desired column.

   The rows are sorted according to the contents of that column.
2. To reverse the order of the rows, click in the same column again.

Column Descriptions There are many columns of data available. Most of these apply only to the implementation rows, not the individual place-and-route runs.

- Implementation<Strategy>: The name of the implementation followed by the strategy (in angle brackets) that implementation is using. Use the check boxes to select which implementations to run. This column cannot be hidden.
- Current Step: The implementation process, such as Synthesis and Map, that is currently running.
- Status: The status of the implementation, such as Running, Pause, and Completed.
- Next Step: The next implementation process that will run after the current process finishes.
- Worst Slack: The worst timing slack for all timing constraints. Negative values indicate timing violations.
- Timing Score: The time, in picoseconds, by which the design is failing to meet the timing constraints. Zero means the run fully met the timing constraints.

  If an implementation has multiple place-and-route runs, the implementation row shows the value of the active place-and-route run.
- Worst Slack (Hold): The worst timing slack for all timing constraints based on hold time. Negative values indicate timing violations.
- Timing Score (Hold): The time, in picoseconds, by which the design is failing to meet the timing constraints based on hold time. Zero means the run fully met the timing constraints.
- Slice: The number of slices used versus the total number available.

- Unrouted Nets: The number of unrouted nets. Anything larger than zero means that place-and-route did not finish.

  If an implementation has multiple place-and-route runs, the implementation row shows the value of the active place-and-route run.

- Run Time: The total run time for the synthesis, map, and place-and-route processes, including multiple place-and-route runs. Times for translate, trace, and exporting files are not included. The format is <*hours*>:<*minutes*>:<*seconds*>.

- Level/Cost: The first number is the Placement Effort Level specified in the Place & Route Design section of the strategy, not necessarily the effort used. See "Placement Effort Level" on page 1148.

  The second number is the number of placement iterations that were run to get a solution. If an implementation has multiple place-and-route runs, the second number is cumulative of preceding runs. That is, if the design only needs one iteration, the list of place-and-route runs shows 1, 2, 3, ...

  The implementation row shows the value of the active place-and-route run.

- Location: The full pathname for the implementation.

- Start: When synthesis started. Not applicable for EDIF projects.

- Level/Cost: The first number is the Placement Effort Level specified in the Place & Route Design section of the strategy, not necessarily the effort used. See "Placement Effort Level" on page 1148.

  The second number is the number of placement iterations that were run to get a solution. If an implementation has multiple place-and-route runs, the second number is cumulative of preceding runs. That is, if the design only needs one iteration, the list of place-and-route runs shows 1, 2, 3, ...

  The implementation row shows the value of the active place-and-route run.

- Description: An editable text box. Initially it just has the implementation name but you can change it.

  To change the description of an implementation, double-click the Description cell. Then enter any text you want.

- GSR: The number of GSR used versus the total number available.

- PIO: The number of PIO used versus the total number available.

- EBR: The number of EBR used versus the total number available.

- PCS: The number of PCS used versus the total number available.

- Number Of Signals: The number of signals in the design.

- Number Of Connections: The number of connections in the design.

Selecting an Implementation and Run

After studying the results, you can select one of the implementations and one of its runs for further development.

To select an implementation and run:

1. Right-click on the implementation's row and choose **Set As Active**.

   This implementation becomes the active one for the project. The row changes to bold type and, in File List view, the implementation is expanded. The previously active implementation becomes inactive.

2. If the implementation has multiple runs, right-click the preferred run and choose **Set As Active**.

   The results of this run will be used in further development such as exporting files. The row changes to italic type.

Finding Results

After you load a design in Diamond, you can find the information you need via the following ways.

- In the active Reports view, choose **Edit > Find**. You can type the desired text into the Find field at the bottom-left of the Reports view window. The first occurrence of the desired text will be found and highlighted in the right side of the window for you. Click **Next** or **Previous** to find more. And check the **Case Sensitive** option if needed for the search. While typing in the text, the Find field will be automatically colored if no occurrence of the text is found.
- In the active Source Editor, after choosing **Edit > Find**, you will get the Find and Replace dialog box. You can enter the text you want to search in the Find What field and start a search. Use the **Find Next** command to find more. If you want to replace the current find, you can use the Replace tab of the dialog box. Check **Match Case**, **Match Whole Word**, **Search Up**, **Regular Expression** options as needed. Use the **Replace** or **Replace All** command to replace the text found.
- If you want to find information without loading the files, you can choose **Edit > Find in Files** from Diamond main window. In the pop-up Find In Files dialog box, type the text you want to find, specify the search path and search filters, and check the desired options: **Search subdir**, **Include hidden files**, **Match case**, **Match whole word**, and **Regular expressions**. Press **Find**. The results will be displayed in the Find Results frame. Double-click any of the findings from the Find Results frame to open the associated source file in the associated editor. For example, if the finding is in a log file, the log file will be opened in the Reports view with the first finding appears on the first line.
- You can search the Output log by clicking in the Output view (in the text, not on the tab) and then pressing **Ctrl-F**. This opens a basic text search dialog box at the top of the Output view. You can type the text in the Find text field and start a search.

Find Results View The Find Results view may not be displayed automatically in your Diamond main window. To turn on the Find Results view, select **View > Show Views > Find Results**. A check mark indicates the frame is displayed.

The Find Results view can be detached from the main window by clicking the detaching icon on the upper-right corner of the view. After detaching, you can double-click on the title bar of the view to get it back to the main window.

See Also ▶"Viewing Logs and Reports" on page 101

- "Navigating Errors and Warnings" on page 103
- "Finding Text in Logs and Reports" on page 103

Viewing Logs and Reports

Lattice Diamond generates log files for all project activities. The log files contain processing information, as well as error and warning messages. If you run processes, reports are generated.

Viewing Logs A log file is displayed in the Output frame as a process is running. A scroll bar can be used to scroll up and down in the information.

Errors are displayed in red. Warnings are displayed in orange. There are also information messages. These messages are also displayed in the Warning, Error, and Info views. These views may not automatically be visible in your Diamond main window. To turn on the views, choose **View > Show Views >** <*view*>. A check mark indicates the view is displayed.

Viewing Reports The Reports view displays reports for the major processes.

There are two panes in the Reports view. The left pane lists the Design Summary information including the report types. The reports in detail are displayed in the right pane.

Type of Report	Description
Project Summary	Lists the summary information of the project including module name, synthesis tool chosen, implementation name, strategy name, target device, device family, device type, package type, performance grade, operating conditions, logic preference file, software product version, project file name, and location.
Process Reports	Lists the synthesis, map, place and route, signal/pad, and bitstream reports in HTML format.
Analysis Reports	Lists the trace and timing reports.
Tool Reports	Lists the I/O SSO analysis, hierarchy parsing, PIO DRC, and ECO Editor reports. Also has a log of Tcl commands used in recent sessions.
Messages	Lists the implementation messages and user defined filters for the messages.

In the Design Summary pane, there is the report icon  . If a report has been generated, the icon appears as  . If the report is not the most recent version,

the icon appears as  . To view the contents of the entire report, click on the report to be viewed. The entire report is then displayed in the right pane of the Reports view. Use the scroll bar to navigate through the report. Some of the reports are divided into sections (for example, Map, Place & Route, and Signal/Pad). Expand the report listing to display the sections in a list. Choose the desired section. The whole report will be displayed with the selected section displayed at the top of the right pane of the Reports view.

You can navigate the reports quickly by using the Find function (right-click in the right pane of the Reports view and choose **Find in Text**).

Other Reports The Synthesize Design stage produces reports that do not appear in the Reports view. You can find these reports in the implementation folder. In the File List view, right-click the implementation name and choose **Open Containing Folder**. A window will open showing the contents of the folder. All of these reports can be read with a text editor.

One of the reports is a detailed description of the device resources that will be used by the design. This report is much more detailed than the synthesis report in the Reports view. The report includes the resources used by each module of the design. Similar information can also be found in the Hierarchy view. For Synplify Pro, look for <*top_module*>.areasrr; for Lattice Synthesis Engine, look for <*top_module*>.arearep.

See Also ▶"Finding Results" on page 100

Cross-Probing from Reports to Schematics

While studying one of the timing or trace reports you might want to see where a module or port is in the design. You can cross-probe, or jump, from the Lattice Synthesis Engine (LSE) timing report and from the place & route trace report to a schematic view of the design.

To cross-probe from the LSE timing report to a schematic view:

1. After running synthesis with LSE, open the Reports view.
2. In the Design Summary column, click **LSE Timing Report**. It's under Analysis Reports.
3. Select text that has the name of one or more module instances or ports of interest.
4. Right-click and choose **Filter in Netlist Analyzer**.

   Netlist Analyzer opens with the technology netlist view of the selected objects. For more information, see "About Netlist Analyzer" on page 58 and "Navigating the Design with Netlist Analyzer" on page 63.

To cross-probe from the place & route trace report to a schematic view:

1. After running place & route of the design, choose **Tools > Synplify Pro for Lattice**.

2. In Synplify Pro, click the **Implementation Directory** tab.
3. Find the **.twr** file and double-click it.

   A text editor opens in Synplify Pro with the report.
4. Find the name of an instance or port of interest and select it.
5. Right-click in the selected name and choose one of the following:
   - **Filter in Analyst** to see the item by itself
   - **Select in Analyst** to find the item in the full schematic
6. If a suitable Analyst view is not open, a dialog box asks if you want to open one. Click **Yes**.
7. Go to the Analyst schematic view to see the item.

Navigating Errors and Warnings

If an error or a warning results from the specific line in an HDL source file, you can easily go to that line to edit the source file.

To navigate errors and warnings:

- In the Reports, Output, Error, or Warning view, double-click the line describing the error or warning.
- In the Reports, Output, Error, or Warning view, right-click the message and choose **Locate in > Text Editor**. If the command is dimmed, there is no link to a source file.

Your default text editor opens with the appropriate HDL source file at the line number specified in the error or warning message. You can then modify the file to debug your design.

See Also ▶"Finding Results" on page 100

Finding Text in Logs and Reports

You can search for text in the Output, Error, Warning, Info, and Reports views.

To search for text in the Output, Error, Warning, and Info views:

1. Right-click in the Output, Error, Warning, or Info view of the Diamond main window and choose **Find in Text**.

   The Find toolbar appears.
2. In the **Find** text box, enter the text that you want to search for. Check the **Case Sensitive** option if you want the text to match exactly as it was entered into the text box. Or, check the **Whole Words** option as desired. As you type each character, the focus in the window changes to where a match is found.

3. Use the **Next** and **Previous** buttons in the Find toolbar to find other occurrences of the text.

To search for text in the Reports view:

1. Right-click in the right pane of the Reports view, and choose **Find in Text**.

   The Find toolbar appears.

2. In the Find text box, enter the text that you want to search for. Check the **Case Sensitive** option if you want the text to match exactly as it was entered into the text box.
3. Use the **Next** and **Previous** buttons in the Find toolbar to find other occurrences of the text.

See Also ▶"Finding Results" on page 100

Filtering Messages

The number of messages produced by the design implementation process can be very large. There are several ways to filter these messages to find the ones of most interest. Messages can be filtered by:

- Implementation process (synthesis, translate, map, place & route, export, or other tool)
- Severity (information, warning, and error)
- ID number
- Text

Filter settings can be saved for use later or to apply to the Output view. Saved settings are available in the Reports view under Messages > User Defined Filters.

Messages are automatically sorted by severity: Error, Warning, and Info. Look for these tabs in the message area of the main window alongside the Output view. If the one you want is not showing, choose **View > Show Views** and select the one you want.

To set filter options:

The message list changes as you set filter options.

1. After running part or all of the implementation process, go to the Reports view. This view opens automatically. If it is not showing, choose **View > Reports**.
2. At the bottom of the Design Summary column, under the Messages folder, select either **All Messages** or one of the user defined filters.

   Messages appear to the right in a spreadsheet display.

3. If filter options are not visible above the spreadsheet, click the Show Filter Selection button. It's the arrow at the top-right of the Reports view.

Several check boxes and buttons appear at the top of the view.

4. Select the processes and severity levels you want to hide.
5. Specify any message ID that you want to hide by right-clicking a message that has that ID and choosing **Filter Messages with This ID**.
6. To hide a specific message, right-click on an example of the message and choose **Filter Messages Exactly Like This**. This hides all messages with the same process, severity, ID, and text. The text must be completely the same. Messages of the same type but using a different signal or module name will show.

   To see messages hidden by this option, click **Filter Details** in the Filter Selection area or right-click in the Message area and choose **Filter Details List**. The Detailed Filter List dialog box opens showing just this set of hidden messages.
7. To stop hiding messages hidden by the Filter Messages with This ID and the Filter Messages Exactly Like This options, open the Detailed Filter List dialog box (above). Clear the **Filter Enable** check box of the desired messages and click **Apply**.

Note

You can also select options by right-clicking a message of the type that you want to hide and choosing one of the Filter Messages commands. These commands hide all messages with the same process, severity, or ID as the selected message.

To undo all filter settings:

1. Click **Reset**.

   A dialog box opens.
2. Do one of the following:
   - If there have been changes, you are offered the option of saving the settings. If you choose to save, see the following procedure.
   - If there have been no changes, click **OK** in the dialog box.

To save filter settings:

1. If the filter options are not visible above the spreadsheet, click the Show Filter Selection button. It's the arrow at the top-right of the Reports view.
2. Click **Save Filter**.

   The Save Filter dialog box opens.
3. Enter a name for the filter.
4. Click **OK**.

   The new filter's name appears in the Design Summary column under Messages > User Defined Filters.

To apply user defined filters to the Output view:

Filter settings must be saved before they can be applied to the Output view.

1. Before running part or all of the implementation process, go to the Reports view. If it is not showing, choose **View > Reports**.
2. At the bottom of the Design Summary column, under the Messages folder, expand **User Defined Filters**.
3. Right-click the desired filter and choose **Apply "***<filter>***" to Output View**.

   A filter icon is placed by the filter's name.
4. Run the desired implementation processes.

   The Output view is loaded as usual except that the specified messages are missing. To see the missing messages, look in the Error, Warning, or Info tabs or select All Messages in the Reports view.
5. To stop using the filter on the Output view, right-click the filter name and choose **Apply "***<filter>***" to Output View** again.

   The Output view does not change, but will display all messages from future processing.

Changing Warnings to Errors

There may be warning messages that you consider to be serious problems and want to stand out more. You can change these warnings to show the error severity level in your design project. You can also save these "promotions" to use in other design projects.

Promotions only show for messages from future actions. Promotions do not affect the display of messages from past actions.

To change a warning to an error in the Reports view:

1. In the Reports view, go to the Messages section.
2. Right-click on the message that you want to promote and choose **Promote Messages with This ID**.

   Not all warning messages can be promoted so this command may be dimmed.

To change a warning to an error without the Reports view:

1. Choose **Project > Message Promotion**.

   The Message Promotion dialog box opens.
2. In the IDs box, type the ID number of the messages that you want to promote. Separate multiple ID numbers with semi-colons (**;**).
3. Click **Promote**.

   Not all warning messages can be promoted so some ID numbers may be rejected.
4. When you are finished in the Message Promotion dialog box, click **Close**.

To restore a message ID to the warning level:

1. Choose **Project > Message Promotion**.

   The Message Promotion dialog box opens.

2. Select the message that you want to restore to the warning level.
3. Click **Remove**.
4. When you are finished in the Message Promotion dialog box, click **Close**.

To save the list of promoted messages for another project:

1. Choose **Project > Message Promotion**.

   The Message Promotion dialog box opens.

2. Click **Export**.
3. In the Export Promotion dialog box, browse to where you want to save the list.
4. Enter a file name.
5. Click **Save**.

   The list is saved to a file with a ".pmt" extension.

6. When you are finished in the Message Promotion dialog box, click **Close**.

To use an existing list of promoted messages:

1. Choose **Project > Message Promotion**.

   The Message Promotion dialog box opens.

2. Click **Import**.
3. In the Import Promotion dialog box, browse to the list.
4. Click **Open**.

   The list of messages appears in the Message Promotion dialog box.

5. When you are finished in the Message Promotion dialog box, click **Close**.

Getting Help for Messages

Some messages have a link to more information in the online Help. This information may help to understand what the message is about or how to fix the problem.

To check for help on a message:

1. Right-click on the message in the Reports, Error, Warning, or Info view.

   A menu appears. The last item in the menu is Help. If it is dimmed, there is no link.

2. If the Help command is available, choose it.

   The Help opens with the appropriate topic.

Chapter 3

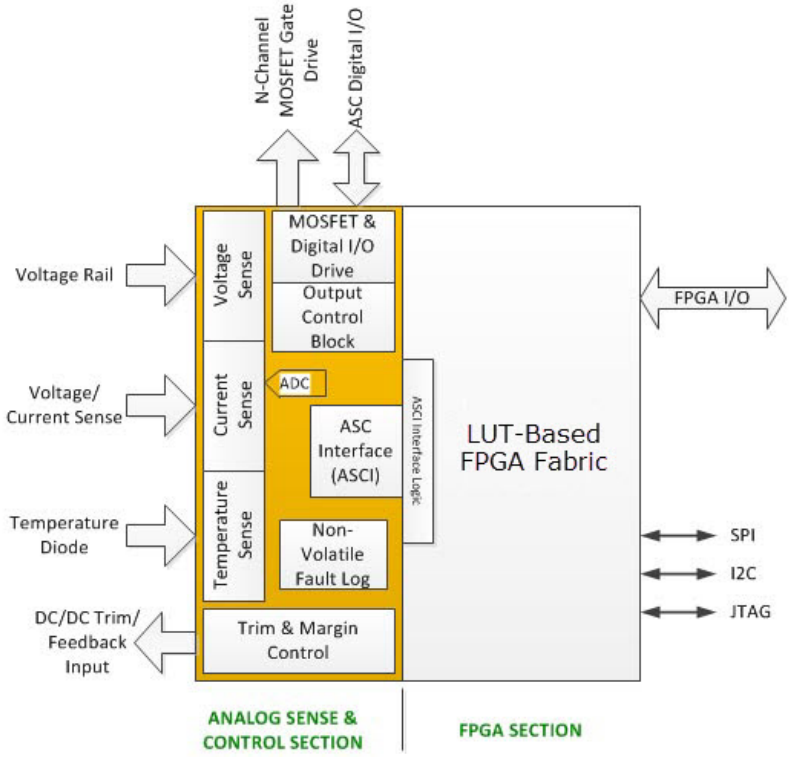
Designing with Lattice Diamond Platform Designer

Included with the Lattice Diamond software is the Platform Designer tool which enables you to build and control a complete hardware management system. With Platform Designer, you can build and control:

- Platform Manager 2
- Platform Manager 2 in a two-device (LPTM21 and LPTM21L) configuration
- Certain devices in the MachXO2HC/HE family with external Analog Sense and Control (ASC) devices
- Certain devices in the MachXO3D family with external ASC devices.
- Certain devices in the MachXO3LF family with external ASC devices.
- Certain devices in the ECP5U/UM family with external ASC devices.

The larger LPTM21 device can configure both ASC and/or LPTM21L as slave devices.

Figure 2: Platform Manager 2 Block Diagram



As shown in the previous figure, the Platform Manager 2 device is made up of an ASC block and an FPGA block. These two blocks provide the necessary programmable building blocks to enable integration of various combinations of hardware management functions into a single chip. The hardware management algorithm is implemented within the FPGA block of the Platform Manager 2. The monitoring and control resources can be scaled up, to meet application requirements, by adding an external ASC to the Platform Manager 2. Each of the ASCs provides additional monitor and control of up to ten voltage rails, two temperature sensors and two current sensors. The overall platform management algorithm that uses the inputs from the external ASC device(s) is located within the FPGA block of the Platform Manager 2. The following diagram shows how an ASC device can be connected to a Platform Manager 2 device to implement a hardware management circuit that monitors the board voltage rails (V), supply currents (I), and board temperatures (T).

Figure 3: ASC Device Connected to a Platform Manager 2 Device

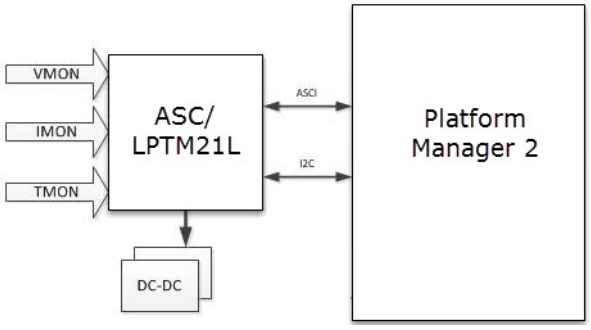
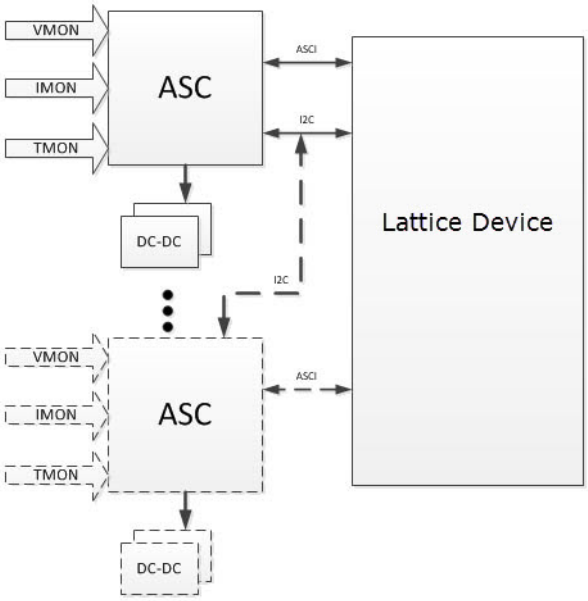


Figure 4: Connecting Multiple ASCs to Lattice Device



Multiple ASC devices can also be connected to imilar to the Platform Manager 2, to offer wider flexibility and control when implementing a complete hardware management system. Devices include:

- Certain devices in the MachXO2HC/HEHC/HE family
- Certain devices in the MachXO3D family with external ASC devices
- Certain devices in the MachXO3LF family with external ASC devices
- Certain devices in the ECP5U/UM family with external ASC devices

Unique to Platform Manager 2 is the smaller package LPTM21L device can be also configured as a slave device combining with ASC. Table 2 lists Platform Manager 2, MachXO2HC/HEHC/HE, MachXO3D, MachXO3LF and

ECP5U/UM devices that support ASCs, and also lists the maximum number of ASCs that can be used for each device.

Table 2: External ASCs/LPTM21 supported by Platform Manager 2, MachXO2HC/HEHC/HE, MachXO3D, MachXO3LF, ECP5U/UM

Family	Device	Maximum Number of External ASCs
Platform Manager 2	LPTM21	3 ASC and/or LPTM21L
Platform Manager 2	LPTM21L	3 ASC and/or LPTM21L
MachXO2HC/HEHC/HE	640HC	2
	1200HC	4
	2000HC	6
	4000HC	8
	7000HC	8
	2000HE	6
	4000HE	8
	7000HE	8
MachXO3D	4300	8
	9400	8
MachXO3LF	640E	2
	1300E/C	4
	2100E/C	6
	4300E/C	8
	6900E/C	8
ECP5U/UM	LFE5U-12F	8
	LFE5U-25F	8
	LFE5U-45F	8
	LFE5UM-25F	8
	LFE5UM-45F	8

See Also ▶"Platform Designer User Interface" on page 112

▶ "Platform Designer Flow" on page 114

About Platform Designer

Platform Designer provides an integrated design environment that enables you to configure the device, implement the hardware management algorithm, simulate, assign pins, and finally generate the JEDEC files required to program and configure the device on the circuit board. It also allows you to import other HDL files to integrate other desired functions.

See Also ▶"Launching Platform Designer" on page 116

- "Platform Designer User Interface" on page 112
- "Platform Designer Flow" on page 114

Platform Designer User Interface

Platform Designer contains separate editors for configuring global settings; current, temperature, and voltage monitors; fan controller and fault logger components; ports, nodes, and logic controls. It also provides a view for checking design rules and resources and building the design. Each editor and view is accessed from the Platform Designer menu pane on the left, which is divided into five main sections: Global, Analog, Components, Control, and Build.

Global The Global Configuration view provides global configuration options, including options for each ASC device. The global ASC Options provide access to configuration settings like the I^2C address and other default settings. The Device Options enable you to select the programming interface, clock source, and boot mode.

Components The Components section enables you to configure the fan, fault logging, hot swap, and PMBus Adapter. It includes the following views:

- Fan Controller – The Fan Controller allows you to add a fan, specify the fan interconnections, and configure the control parameters. The fan controller can be configured to work with 2-wire, 3-wire, or 4-wire fans, and up to sixteen fans can be controlled. The fan controller generates a pulse width modulated (PWM) signal to set fan speeds.
- Fault Logger – The Fault Logger enables you to specify and configure fault conditions that are logged to non-volatile memory. You have the option of using standard fault logging that is based on voltage, current, and temperature status; or using the full-featured fault logging.
- Hot Swap – **(Platform Manager 2, MachXO2HC/HEHC/HE, MachXO3D, and MachXO3LF only.)** The Hot Swap component enables you to configure the ASC for a hysteretic closed-loop to limit the in-rush current in an application where power is rapidly applied. Such an application can include such a line card, blade, or a PCB that is plugged into a "hot" backplane.
- PMBus Adapter – **(Platform Manager 2, MachXO2HC/HE, MachXO3D, and MachXO3LF only.)** The PMBus Adapter component allows you to configure the adapter for connecting analog Point of Load (POL) DC-DC converters to PMBus using the ASC and FPGA.

Analog The Analog section enables you to configure the settings for current, temperature, and voltage monitors. It includes the following editors:

- Current – The Current editor enables you to configure the low-voltage and high-voltage current monitors for each ASC device. Platform Designer automatically populates the current trip points available based on the sense resistor selection. You can configure the trip points and other current monitor settings here.
- Temperature – The Temperature editor enables you to configure the temperature monitors (TMON) for each ASC device. Multiple monitor levels can be selected for each TMON, with temperature values displayed in ^{O}C.
- Voltage – The Voltage editor allows you to configure voltage monitoring, trimming, margining, and voltage identification (VID) settings for each ASC device in your system. You can edit multiple settings by using the Voltage Monitor & Control dialog box. To open the dialog box, double-click an active cell in the ASC column or the Pins Monitor/Trim column on any of the three spreadsheets, or right-click the cell and choose **Edit**. You can also edit settings individually on the Voltage spreadsheet or the VID spreadsheet by single-clicking an editable cell.

Control The Control section enables you to configure signals and set the control logic. It includes the following editors:

- Ports & Nodes – The Ports & Nodes editor provides spreadsheets for viewing the internal nodes and configuring external signals (ports) of the chip set. The Ports sheet shows all physical FPGA-based PIOs that are available for use as external connections to the design and allows you to assign labels to logical ports, specify the I/O and register type, and group ports of a common type to form a bus. The Nodes sheet shows internal signals already generated to connect the implemented features of the design. The nodes sheet allows you to create additional signals for connecting different features on-chip. The GPIO sheet allows you to configure and label the ASC-based general purpose I/Os. The HV Outputs sheet allows you to label and configure the high-voltage output ports.
- Logic – The Logic editor enables you to create sequences, exception logic, supervisory equations, and timers for one or more state machines. It also allows you to import modules into your design from HDL code. The Sequence page of the Logic editor allows you to add state machines and gives you access to the Multiple State Machines dialog box for managing the state machine library.

Build The Build view enables you to check the design status, examine the amount of hardware resources consumed, run DRC check, compile the design, and generate the merged JEDEC file. The Build view includes a summary table of enabled components and programming options, and it shows the amount of hardware resources of the device that have been consumed by the current design configuration. The Build view enables you to generate a test bench, including stimulus, for simulating your design. It also allows you to export a detailed configuration report.

See Also "Working with Platform Designer Editors" on page 120

Platform Designer Input Files

The input files for Platform Designer include the Platform Designer project file (.ptm) and HDL source files (.v, .vhd).

The .ptm file holds all of the Analog Sense and Control settings, as well as the logic and port information, for the active Platform Designer project. A .ptm file is generated automatically for a new project. You can also add a new or existing .ptm file, which allows you to choose from multiple project files in a single implementation.

HDL source files are not required, but you can add them to the Diamond project. Once the HDL files are added to the Diamond project, you can use Platform Designer's Logic Editor to import an HDL module into the Platform Designer project.

See Also
- ▶See "Working with Platform Designer Project (.ptm) Files" on page 118
- "Importing HDL Modules" on page 157.
- "Creating a New Platform Designer Project" on page 116

Platform Designer Output Files

Platform Designer automatically generates the background HDL files when you compile the design. The "Generate Jedec" process produces a merged JEDEC of the FPGA and ASC JEDEC files for programming. ECP5U/UM output file is .bit.

See Also ▶"Building the Design" on page 157

Platform Designer Flow

The number of tasks involved in the Platform Designer flow will vary, depending on the selected Platform Manager 2, or MachXO2HC/HE, MachXO3D, or MachXO3LF, or ECP5U/UM, the number of external ASC/ LPTM21L (for LPTM21/LPTM21L as masters, LPTM21L and/or ASC can be slaves) devices selected, and the types of configuration needed to complete the design. The following outline shows a typical flow of required and optional tasks.

1. Create a project in Lattice Diamond. Select the base device and package; Platform Manager 2, or MachXO2HC/HE, MachXO3D, or MachXO3LF, or ECP5U/ECP5UM. Then add the number of external ASC devices needed. For Platform Manager 2 devices LPTM21 / LPTM21L selected as master devices, LPTM21L and/or ASCs can be combined as slave devices.
2. Specify global settings:
   - ASC options, including reset type, and default options for voltage, current, and temperature

- device options, including programming interface, clock source, and boot mode

3. Configure the ports and nodes:
   - ports and port groups
   - internal nodes
   - general-purpose I/Os
   - high-voltage outputs
4. Configure the analog settings:
   - Configure the voltage monitors (VMONs):
     - schematic net names and parameters
     - trim/margin parameters
     - voltage identification (VID) tables
   - Configure the current monitors (IMONs).
   - Configure the temperature monitors (TMONs).
   - Configure one or more fans:
     - fan type and pulse width modulation
     - fan interconnections
   - Configure PMBus Adapter.
   - Enable and configure fault logging.
5. Define the platform management logic, including sequencing and monitoring:
   - timers
   - sequencer instructions and exceptions for one or more state machines
   - supervisory logic equations
   - modules that are imported from HDL
6. Build the design.
   - Examine the resources consumed.
   - Run design rule check.
   - Compile the design to generate the HDL code and synthesize the logic.
   - Assign Pins in Spreadsheet View.
   - Generate the JEDEC to process the design and generate the merged JEDEC file.
7. Simulate the design.

See Also
- "Platform Designer User Interface" on page 112
- "Creating a New Platform Designer Project" on page 116
- "Working with Platform Designer Project (.ptm) Files" on page 118

Launching Platform Designer

You can launch Platform Designer by creating a new project in Diamond that targets a Platform Manager 2 LPTM21/LPTM21L device with external ASCs and/or LPTM21L devices, or a MachXO2HC/HE with external ASCs, MachXO3D with external ASCs, or MachXO3LF with external ASCs, or an ECP5U/UM with external ASCs. You can also launch Platform Designer by opening an existing Platform Designer project file (.ptm).

Platform Designer is also available from the Diamond Tools menu when you open an existing project that targets a Platform Manager 2 LPTM21/LPTM21L device with external ASCs and/or LPTM21L devices, or a MachXO2HC/HE with external ASCs, or a MachXO3D with external ASCs, or a MachXO3LF with external ASCs, or an ECP5U/UM with external ASCs.

See Also

- "Creating a New Platform Designer Project" on page 116
- "Opening an Existing Platform Designer Project" on page 118
- "Working with Platform Designer Project (.ptm) Files" on page 118
- "Working with Platform Designer Editors" on page 120

Creating a New Platform Designer Project

When you create a new project that targets the Platform Manager 2, or a MachXO2HC/HE with external ASCs, or a MachXO3D with external ASCs, or a MachXO3LF with external ASCs, or an ECP5U/UM with external ASCs, Diamond generates a new Platform Designer project file (.ptm) and opens the Platform Designer user interface.

To create a new Platform Designer project in Diamond:

1. In Diamond, choose **File > New Project** to open the Create a Lattice Diamond Project wizard.
2. Click **Next**.
3. In the Project section, type a name for the project and use the Browse button to navigate to a location.
4. Click **Next**.
5. The Add Source section is an optional step that allows you to add HDL source files to the project or an existing Platform Designer project (.ptm) file. If you add an existing .ptm file, the default .ptm file will not be added to the project.

   To skip the Add Source step, click **Next**.

   To add source files to the project, click the **Add Source** button to navigate to the files and select them, and then click **Next**.
6. In the Select Device dialog box, do one of the following:

- Select **Platform Manager 2** in the Family section, and then select the desired LPTM device from the Device section.

  Platform Manager 2 devices contain one embedded Analog Sense and Control (ASC) device. There are 2 devices in the Platform Manager 2 family, LPTM21 and LPTM21L in a smaller package configuration. The LPTM21/LPTM21L device allows you to add up to three external ASC and/or LPTM21L devices. The LPTM20 device allows you to add one external ASC device. To add an external ASC and/or LPTM21L device, select **LPTM21/LPTM21L**. The “Select ASC Device” section will then appear near the bottom of the dialog box. You will note a 3 device radial button selection for mixing and matching selection of ASC and LPTM21L as slave devices.

- Select **MachXO2HC/HE** in the Family section, and then select the desired LCMXO2 HC device—640HC or larger—from the Device section.

  The “Select ASC Device” section appears near the bottom of the dialog box. Up to eight ASC devices can be added, depending upon the size of MachXO2HC/HE device.

- Select **MachXO3D** in the Family section, and then select the desired from the Device section.

  The “Select ASC Device” section appears near the bottom of the dialog box. Up to eight ASC devices can be added, depending upon the size of MachXO3D device.

- Select **MachXO3LF** in the Family section, and then select the desired from the Device section.

  The “Select ASC Device” section appears near the bottom of the dialog box. Up to eight ASC devices can be added, depending upon the size of MachXO3LF device.

- Select **ECP5U/UM** in the Family section, and then select the desired LFE5UM device—25F or 45F—from the Device section.

  The “Select ASC Device” section appears near the bottom of the dialog box. Up to eight ASC devices can be added.

7. In the “Select ASC Device” section, select the number of ASC devices you would like to include.

   The package type, performance grade, operating conditions, and part name are all tied to the device selection.

8. Click **Next**.

   In the Select Synthesis Tool dialog box, select the synthesis tool that you want to use. The choices are **Lattice LSE** and **Synplify Pro** and can be changed at any time.

9. Click **Next**.

   In the Project Information dialog box, the option “Create Platform Designer file” is selected by default. This will create a .ptm file using the project name shown in the File name box and cause Platform Designer to open. If

desired, you can assign a different name for the .ptm file and select a different directory location.

Note

If you added an existing .ptm file to the project in Step 4, the “Create Platform Designer file” option will not be available.

10. Click **Finish**.

    Platform Designer opens within Diamond and loads the .ptm file.

See Also ▶“Opening an Existing Platform Designer Project” on page 118

▶ “Working with Platform Designer Project (.ptm) Files” on page 118

Opening an Existing Platform Designer Project

After you have created a project in Diamond that targets a Platform Manager 2, or a MachXO2HC/HE with external ASCs, or MachXO3D with external ASCs, or MachXO3LF with external ASCs, or an ECP5U/UM with external ASCs, you can open the Platform Designer project file (.ptm) for the active implementation.

To open an existing Platform Designer project in Diamond:

1. In Diamond, choose **File > Open > Project**, and navigate to the Lattice Diamond file (.ldf) for the Platform Designer project.

   Diamond loads the file and shows the .ptm file in the File List view.

2. Do one of the following:

   ▶ Click the Platform Designer button  on the toolbar or from the Tools menu.

   Platform Designer opens and loads the active .ptm file

   ▶ In the File List view, expand the Input Files folder and double-click the .ptm file that you want to open.

   Platform Designer opens and loads the .ptm file.

See Also ▶“Working with Platform Designer Project (.ptm) Files” on page 118

Working with Platform Designer Project (.ptm) Files

The Platform Designer project file (.ptm) holds all of the Analog Sense and Control settings, as well as the logic and port information, for the active implementation. When you create a new Platform Designer project, the .ptm file is generated automatically by Diamond. You can also add a new or

existing .ptm file to your project. The project's .ptm files are maintained in the Platform Design Files folder in the File List view for the active implementation, and the active .ptm file in this folder is displayed in bold. Only one .ptm file can be active for an implementation.

To add a new .ptm file to a Platform Designer project:

1. In the File List view of Diamond, right-click the name of the active implementation, and choose **Add > New File**.
2. In the dialog box, select **Source Files** from the Category list, and then select **Platform Designer File**.
3. Type a name for the new .ptm file. If desired, select a different location for the file by clicking the Browse button.
4. Click **New**.

   Diamond loads the new .ptm file and sets it as the active Platform Designer file for the current implementation.

To add an existing .ptm file to a Platform Designer project:

1. In the File List view of Diamond, right-click the name of the active implementation, and choose **Add > Existing File**.
2. In the "Files of type" list, select **Platform Designer Files** from the drop-down menu.
3. Browse to the location of the desired .ptm file, select it, and click **Add**.

   Diamond loads the .ptm file and sets it as the active Platform Designer file for the current implementation.

To change the active/inactive status of a .ptm file:

1. If the .ptm file that you want to make active or inactive is still open in Platform Designer, close it.
2. Right-click the .ptm file and choose the appropriate command: **Set as Inactive** or **Set as Active**.

Opening an Inactive .ptm File Platform Designer allows you to open an inactive .ptm file to examine the configurations. The inactive file will open in read-only mode, which will enable you to examine each view but not allow any changes. Platform Designer will close the active .ptm file, if it is open, before opening the inactive one. It will also prompt you to save any changes. You will not be able to execute any of the operations in the Build view (such as Compile, Generate JEDEC) when viewing an inactive .ptm.

Saving a .ptm File When you save an active .ptm File, Platform Designer takes all of the configuration changes that are in memory and saves them to the .ptm file. When you use the "Save as" command to save the.ptm file with a different name or to a new location, Platform Designer includes all of the collateral files for the implementation, as well as any changes in memory. You can also use the "Save as" command for an opened inactive .ptm file.

See Also ▶"Creating a New Platform Designer Project" on page 116

▶ "Platform Designer Flow" on page 114

Working with Platform Designer Editors

The Current, Temperature, Voltage, and Ports & Nodes editors in Platform Designer use a format that is very similar to a spreadsheet. The columns for these editors are arranged in the order of the most commonly used settings. For example, in the Voltage editor, the four most commonly used columns—ASC Device, Pins Monitor/Trim, VMON Schematic Net Name, and Nominal Profile 0—are positioned first and are locked from horizontal scrolling so that they are always visible for association with other columns and for easy access. The cells of columns that are locked from horizontal scrolling are colored with a light gray background. Columns for less frequently used settings, such as Window Mode, are positioned further to the right and are not locked from horizontal scrolling. The cells of these columns have a white background.

Platform Designer editors differ in a couple of ways from standard spreadsheet formats. In Platform Designer spreadsheets, when you click a cell to enter a value, you click only once and then choose a selection from the drop-down menu or type a value, depending on the type of cell selected. Because of this, navigating from cell to cell by keyboard is slightly different, and copying or cutting and pasting from multiple cells is not supported. The following sub-topics explain how to use the features of Platform Designer editors.

Sort by Columns You can sort the row order by a single column by clicking the column heading. An up arrow in the heading indicates that the rows are sorted in ascending order. Click the heading again to sort the list in descending order.

In the Voltage, Current, and Temperature editors, you can also sort the rows by multiple columns.

To sort by multiple columns:

1. Choose **View** >  **Sort t**o open the Sort by Column(s) dialog box.
2. Select the first column to sort by from the "Sort by" drop-down menu. Click Ascending or Descending to control the order of the sorted rows.
3. Use the "Then by" boxes to select additional columns, as well as the Ascending/Descending option for each column. Click **More** to add more "Then by" menus if needed.

   The rows will be sorted first by the column specified in the "Sort by" box and then by the additional columns in the "Then by" boxes in sequence.
4. Click **OK**.

To return to the default layout:

- Right-click any column heading and choose **Default Layout** option. This returns the sorted order to the original view.

Adjust Column Width You can adjust column widths manually or choose the right-click command to fit the column to the text.

To adjust the column width manually:

- Drag the vertical border on the right of a column heading.

To fit the column width to the text, do one of the following:

- Double-click the vertical border on the right of a column heading.
- Right-click the desired column heading and choose **Fit Column**.
- Choose **View > Column > Fit All Columns** to fit all column widths to the text.

Hide and Re-display Columns You can hide a single column in the Voltage, Current, or Temperature editor or select several columns that you want displayed or hidden.

To hide a single column:

- Right-click the desired column heading and choose **Hide Column**.

To hide several columns:

1. Choose **View > Column > Visible Columns** or press Shift+F5 to open the Visible/Hide Columns dialog box.
2. Do one of the following:
   - In the “Show these Columns list,” select the columns that you want to hide, and then click the single-arrow button to move them to the “Hide these Columns” list.
   - Click the double-arrow button to move all columns to the “Hide these Columns” list, and then select those you want displayed by selecting them and moving them to the “Show these Columns” list.
3. Click **OK**.

To re-display some of the hidden columns:

- Press **Shift+F5**, and move the hidden columns back to the “Show these column(s) list.”

To re-display all columns:

- Press **Shift+F5**, and click the double-arrow button to move all hidden columns to the “Show these column(s)” list.

Edit Cell Values The editable cell values in each editor are color-coded blue or black. Blue indicates a default value, and black indicates an edited value. Values that appear in gray are values that cannot be edited, including tool calculated settings or settings that are implied by other areas of the editor.

To edit a single cell value, click once inside the editable cell and enter or select a new value. In the Voltage editor, you can use the Voltage Monitor and Control dialog box to edit the properties for an entire row. See "Setting Multiple VMON/Trim Properties" on page 127.

You can copy or cut a text entry of a single editable cell and paste it into another cell of the same type. You can do this by using the right-click menus or the standard Ctrl+C, Ctrl+X, and Ctrl+V keyboard shortcuts. Copying and pasting multiple cells is not supported.

Right-click a Text-entry Cell to Open the Complete Menu The right-click menu will sometimes vary for text-entry cells, depending on whether you have single-clicked a cell for editing or have highlighted the entire cell. For example, on the Ports sheet, when you single click a Logical Name cell, the text in the cell is highlighted so that you can type a new label. But if you right-click while only the text is highlighted, the menu will display editing commands such as cut, copy, paste, or delete, but it will not include the Add Group or Show In commands.

To access the complete right-click menu for a text-entry cell:

- Right-click a cell without clicking it to highlight the entire cell.
- If you have already single-clicked the cell, press the Enter key to highlight the entire cell.

Navigate from Cell to Cell Because of the convenient one-click access to parameters in the Current, Temperature, Voltage, and Ports & Nodes editors, navigating by keyboard from cell to cell involves an additional small step.

To enable the navigation capability, do one of the following:

- Select a cell in the ASC Device column or the Pin column by single-clicking it.
- If you have already opened an editable cell by clicking it, press **Enter** to highlight the entire cell. If the cell contains a drop-down menu, you might need to press Enter a second time to enable navigation.

Note

If you have single-clicked an editable cell in one of the columns that are locked from horizontal scrolling, you will only be able to navigate among the cells of the locked columns after you press Enter. To enable full navigation afterward, click a cell in the ASC Device column or Pin column, or click an editable cell in a column that is not locked from horizontal scrolling and press Enter.

You can then use the arrow keys to move up and down the cells of a column or across the cells in a row. You can also use the Tab key and the Shift+Tab combination to move forward and back across a row. When you have reached a targeted cell for editing, single-click the cell. To continue navigating afterward, press the Enter key.

Setting Global Options

For a new Platform Designer project, you would normally begin by using the Global Configuration view to configure those features that affect the entire design. This includes device options for operation mode and externally connected components such as dual-boot SPI flash. It includes ASC global options as well as options for each individual ASC of the target chip set.

See Also

- "Setting ASC Options" on page 123
- "Setting Device Options" on page 124
- "Configuring a SPI Flash Model" on page 125

Setting ASC Options

The ASC options are divided into global ASC options, which affect the entire project, and specific ASC options that affect only a given ASC. Use the left portion of the ASC Options to set the values for each individual ASC, and use the Global ASC Options on the right to set global ASC values.

Options for Each ASC The following options are available for individual ASCs:

- ASC Name – labels the ASC with a unique name that will identify it globally.
- CLT Rate – sets the closed loop trim update rate, in µs or ms.
- I^2C Base Address (ASC0 only) – sets the 4 MSB in the 7-bit I^2C address of ASC0. This base 4 MSB is common for all ASCs in the system. This setting is not available for ECP5U/UM based designs. The I^2C base address is locked to the default value.
- I^2C External Resistor – displays the resistor values that are used by the ASC to set the three LSB of the I^2C address. See the ASC data sheet for more details.
- I^2C Address – displays the full address of each ASC, using the configured base address and the 3 LSB that are set by the external resistor.
- UES Bits – sets a 32-bit user electronic signature for storing unique data inside the ASC. This setting is not available for ECP5U/UM based designs.
- Reset Type – configures the reset type as mandatory or optional. For ASC0, the reset type is fixed to mandatory. Mandatory ASCs have their reset signals tied together on the board. If one of these ASCs needs to be reset, the whole system is reset. Optional ASCs are handled individually and have a dedicated reset signal.
- Reset Source – displays the port defined in the reset source for an optional reset type.

Global ASC Options The following global options affect all ASCs:

- EIA Resistor Standard – limits the resistor selection to the EIA standard selected or, if Exact is selected, calculates the exact resistor values. This

value is used by the TRIM calculator in the voltage editor. It determines which resistor values will be recommended by the calculator.

- Open Resistor Threshold – sets the maximum resistor value above which the resistor is treated as an open circuit. This setting is used by the TRIM calculator.
- DC-DC Options
  - DC-DC Library Directory – sets the location for the DC-DC Library. The default directory location is inside the Diamond installation sub-directories.
  - Build DC-DC Library – opens the DC-DC Library Builder Wizard, which enables you to add or edit a DC-DC Converter model.
- VID Options
  - VID Tables Directory – sets the location for the voltage ID tables. The default directory location is inside the Diamond installation sub-directories.
  - Build VID Tables – opens the VID Table dialog box, which enables you to add or edit a VID table.
- Voltage, Current, and Temperature Options

  The Voltage, Current and Temperature monitors are populated with default trip points each time a new project is started. The ASC Global Options Default Trip Point Selections allow you to change the default trip points for each of the monitor types at the start of a new project.

See Also "Configuring Voltage Identification (VID) Tables" on page 132

"Defining DC-DC Converter Models" on page 130

Setting Device Options

Select the Device Options tab of the Global Configuration View to configure the operation mode and the external connected components.

Operation Mode Options The Operation Mode section includes the following options:

- Programming Interface:
  - Platform Manager 2 and MachXO2HC/HE, MachXO3D, and MachXO3LF: JTAG or I^2C
  - ECP5U/UM: JTAG only.
- Background Programming:
  - Platform Manager 2 and MachXO2HC/HE, MachXO3D, and MachXO3LF: None, I2C, or JTAG.
  - ECP5U/UM – JTAG or SPI (Background Programming Interface to SPI Flash).
- 8MHz Clock Source for Device Operation – ASC0 or Global_Clock

External Connected Components The External Connected Components section includes the following options:

- SPI Models Directory – Select a location for the SPI models directory.
- Build SPI Models – Configure a new SPI model and add it to the SPI Models Directory, or edit or delete an existing SPI model.
- Boot Mode – Normal or Dual Boot. When Dual-Boot is selected, the SPI model option will appear. Dual-boot requires that a SPI flash be specified. ECP5 only supports Normal mode - ECP5 dual boot is configured through Diamond Deployment Tool.
- ASC I^2C Write Feature – This feature allows you to configure the I^2C write access to the ASC. This feature is locked to Enabled in the ECP5 solution.
  - When “Disabled” is selected, it prevents writing to the ASC on-chip registers over the I^2C bus.
  - When “Enabled” is selected, it allows writing to the ASC on-chip registers over the I^2C bus.
  - When “Controlled by ASCx_GPIO1” is selected, the ASCx_GPIO1 input is used to control whether I^2C write access is enabled.
  - FPGA Port (External Connection) – This option appears when “Controlled by ASCx_GPIO1” has been selected. You will need to select the FPGA port that will connect to the GPIO1 on all ASCs. Afterwards, the GPIO1 will be changed to “IN” type for each ASC.

See Also “Defining DC-DC Converter Models” on page 130

“Configuring a SPI Flash Model” on page 125

Configuring a SPI Flash Model

You can use the Global Configuration View to add and configure a new SPI flash model or edit a current one. The SPI flash models are saved in the SPI Models directory. The default directory location is inside the current project sub-directories.

To configure a SPI flash model:

1. In the Global Configuration View, select the Device Options tab.
2. In the External Connected Components section on the right, do one of the following:
   - Click **Add/Edit SPI Model** in the Value column of SPI Options.
   - If you are using Dual Boot mode, select **Add/Edit SPI Model** from the Dual-Boot SPI Flash pop-up menu in the Value column.

   The SPI Models Configuration dialog box opens and displays the library of currently available SPI models.
3. Do one of the following:
   - To edit an existing SPI model, select it from the list on the left.

- To add a new SPI flash model, click **New**, and then select the newly created model from the list.

4. Optionally, give the selected model a unique name, by single-clicking the name and typing a new one.
5. Select the desired values for the Flash Configurations and Flash Opcodes by double-clicking the value cell and choosing from the menu, and then press enter. Click **Apply** to save one or more selected values or to save all the values for a selected model.
6. If desired, select a different SPI flash model from the list, and repeat Steps 4 and 5.
7. When all options have been set for all the models you are configuring, click **OK**.

See Also "Setting ASC Options" on page 123

"Setting Device Options" on page 124

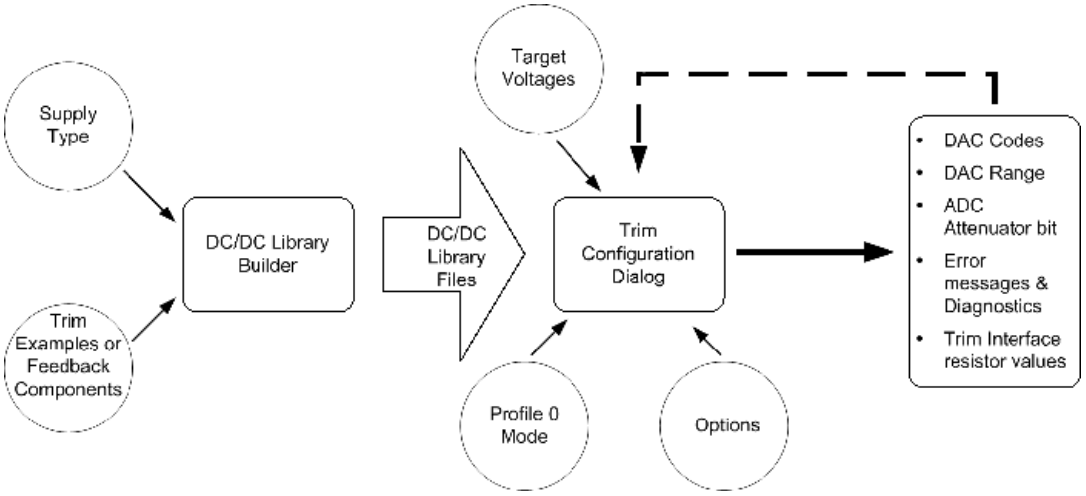
Configuring Voltage Sense (VMON) and Control (TRIM)

The Voltage editor enables you to manage voltages and settings related to voltage monitors (VMONs) and trim channels in the Platform Manager 2 or external ASC devices. You can edit a voltage property by clicking the desired cell on the Voltage spreadsheet, or you can use the Voltage Monitor and Control dialog box to edit multiple properties. Double-clicking a VMON/Trim cell in the Pins Monitor/Trim column opens the Voltage Monitor and Control dialog box. From the Voltage Monitor and Control dialog box, you can access the DC-DC Converter Library and the VID Table library.

Setting up the trim and margin capabilities involves the DC-DC Library Builder, which is used to define the voltage adjustment characteristics of the power supply or supplies that you wish to use. The Trim/Margin portion of the Voltage Monitor and Control dialog box is then used to configure each trim channel for the desired power supplies and output voltages. The following diagram illustrates this flow:

Modifications to existing DC/DC library files must be made from within the DC/DC Library Builder Wizard. The DC-DC Converter selection, as well as changes in the desired target voltages, can be made in the Trim/Margin section of the Voltage Monitor and Control dialog box. It is strongly advised that library files not be modified while trim cells are being configured, because this can create confusion and make it difficult to detect errors in your work. If a "discrete" supply is to be used at several different voltages, a separate library entry for each unique output voltage should be created.

The Trim/Margin dialog enables you to recalculate the trim with a minimum amount of parameter re-entry. Target voltages need to be re-entered only if the desired supply type is changed.



See Also ▶"Defining DC-DC Converter Models" on page 130

▶ "Setting VMON Properties" on page 128

Setting Multiple VMON/Trim Properties

The easiest way to set multiple properties for a selected voltage monitor (VMON) or high-voltage monitor (HVMON) is to use the Voltage Monitor & Control Properties dialog box. The edited settings will then be displayed in the Voltage editor.

To access the Voltage Monitor & Control Properties dialog box:

1. Select the tab in the Voltage editor for those settings that you want to configure.
2. In the Pins Monitor/Trim column, double-click the VMON that you want to configure. Alternatively, right-click and choose **Edit** from the pop-up menu.

   The Voltage Monitor & Control Properties dialog box opens and displays the Voltage, Trim/Margin, or VID section, depending on the tab you selected in the Voltage editor.

The number of tabs available in the Voltage Monitor & Control Properties will depend on whether the selected VMON supports an associated TRIM pin. Most of these properties can also be defined individually on the Voltage spreadsheet.

See Also ▶"Setting VMON Properties" on page 128

▶ "Setting Trim and Margin Properties" on page 129

▶ "Configuring Voltage Sense (VMON) and Control (TRIM)" on page 126

Setting VMON Properties

The Voltage section of the Voltage Monitor & Control Properties dialog box allows you to specify the nominal voltage and the logical names and trip points for each comparator.

- Schematic Net Name – This can consist of any combination of alphanumeric characters.
- Nominal Voltage – This defines both the nominal voltage and the closed loop trim target value for voltage profile 0. If the value is 5.7V or more and the monitor is not an HVMON, the voltage will be used to calculate an external resistor divider and select an associated trip point. A negative voltage will calculate a divider reference as well as the resistor divider. Acceptable values are from -100 volts to 100 volts.
- Logical Name – The logical names, for comparators A and B, are used in the logic builder to control the sequence and equations based on the VMON status.
- Trip Point Selection – A specific VMON trip point, for comparator A and comparator B, is associated with each logical name.
- 64 µs Glitch Filter – When selected, this option configures the comparator so that supply glitches narrower than 64 microseconds are ignored. The output will transition only if the changed status remains active for a period longer than 64 microseconds. If this option is not selected, the comparator output will toggle within 16 microseconds from the time the voltage transitions through the appropriate trip point.
- Window Mode – To use the window mode, the Comparator B threshold should be lower than the threshold setting of comparator A. The window mode output will replace the comparator A output. The window output is logical high if the Comparator B output is high and the Comparator A output is low (i.e., the voltage monitor reading is between Comparator A and Comparator B).
- External Resistor Divider – This section is used for monitoring a voltage whose highest trip point exceeds 5.70V or is a negative voltage. These values are calculated automatically when the nominal voltage specified is negative or exceeds 5.70V. You can overwrite these values to work with your preferred components.
  - Rsupply – Valid value range is 10 ohms to 10 M ohms
  - Rground – Valid value range is 10 ohms to 15 K ohms. The Rground maximum value is restricted to maintain the accuracy of the voltage monitors.
  - Divider Reference – Valid value range is 2.5 V to 6 V. The divider reference is used when monitoring voltages below ground. The external divider is connected between the negative supply being monitored and a positive reference supply.

See Also ▶"Configuring Voltage Sense (VMON) and Control (TRIM)" on page 126

Setting Trim and Margin Properties

The Trim/Margin section of the Voltage Monitor and Control dialog boxenables you to configure the trim cell operating mode and desired output voltages measured at a voltage monitor (VMON).

- Trim Schematic Net Name – This can consist of any combination of alphanumeric characters.
- Trim Configuration Mode – The Trim Calculator, which is selected by default, calculates resistor interface and DAC values from the target voltages and DC-DC converter.

  Manual trim configuration mode allows you to select the CLT loop polarity and to manually enter the DAC code for each voltage profile.
- DC-DC Converter – Select a DC-DC converter model from the drop-down menu. The menu includes all of the models that are in the DC-DC Library. To add a new model, scroll down and select **Add/Edit DC-DC** to open the DC-DC Library Builder Wizard.
- CLT Loop Polarity – If you selected Manual for the trim configuration model, select the polarity for closed loop trim.
- Voltage profile – Enter a desired output voltage for each profile. These profiles are used, along with the nominal voltage profile (profile 0), as the targets for the trim calculator.
- DAC Output Range (BPZ Voltage) – If you are using the Trim Calculator, the bipolar zero output voltage range will be calculated automatically. If you are using manual trim configuration mode, select the value from the drop-down menu.

If you have selected Trim Calculator for the trim configuration mode, click the **Calculate** button after you have made your selections. The dialog box will then provide the following information:

- the resistor values needed to interface the DAC to the TRIM pin in the DC-DC converter
- the voltage that will be achieved with each of the voltage profiles
- the DAC code needed to produce each of the desired output voltages

If a profile target voltage is outside an achievable range, it will be tagged with *Error. To resolve errors, click the **Error Details** button and follow the instructions for obtaining a possible solution.

Setting Trim Configuration Options Use the Trim Configuration Options dialog boxto adjust settings and to help resolve a target voltage error.

Click the **Options** button in the Trim/Margin Properties dialog box to set the following options:

- Maximum DAC Code Range – This setting and the Max Supply Adjustment Range work together to define the sensitivity of the trim system. The value entered defines the DAC value required to trim the supply voltage by the amount specified in the Max Supply Adjustment Range box.

- Max Supply Adjustment Range – This value defines how much the power supply output voltage will change when the DAC code is at the value in the Maximum DAC Code Range box.
- Attenuation Crossover Voltage – If the voltage in Voltage Profile 0 is greater than the voltage in the Attenuation Crossover Voltage text box, the DAC input attenuator will be enabled. The attenuation crossover voltage should be selected so that the attenuation bit is turned on if any portion of the trim range is over 2.048 volts. The default value of 1.9 volts is appropriate in most cases.
- Vbpz Selection – When Auto is selected, the bi-polar zero output voltage is calculated automatically to reach all the target voltages. Selecting one of the other voltages (0.6V, 0.8V, 1V, or 1.25V) can sometimes reduce the number of resistors.

See Also ▶"Configuring Voltage Sense (VMON) and Control (TRIM)" on page 126

Defining DC-DC Converter Models

The DC-DC Library Builder Wizard enables you to define the voltage adjustment characteristics of DC-DC converters and voltage regulators. This information is stored in a library so that any given model of DC-DC converter can be used in multiple trim cells or Platform Designer projects without having to re-enter the information each time.

You can add new DC-DC converter models to the library, delete models, or edit the parameters of a model at any time by launching the wizard. Any converter model in the library can be selected in the Trim/Margin portion of the Voltage Monitor and Control dialog box.

You can launch the DC-DC Library Builder Wizard from the Global Configuration view or from the Voltage Monitor and Control dialog box.

To launch the DC-DC Library Builder Wizard from the Global Configuration View:

1. In the Global Configuration view, select the ASC Options tab.
2. In the DC-DC Options section of Global ASC Options, click **Add/Edit DC-DC**.

To launch the DC-DC Library Builder Wizard from the Voltage Monitor and Control dialog box:

1. In the Voltage view, select the Trim/Margin tab.
2. Double-click a cell in the DC-DC Converter column or a VMON/Trim cell in the Pins Monitor/Trim column.
3. In the Voltage Monitor and Control dialog box, click the DC-DC Converter drop-down menu.
4. Scroll down and choose **Add/Edit DC-DC**.

The DC-DC Library Builder Wizard provides the option of creating a new entry in the library (using the New button) or modifying an existing entry. The flow for both operations is similar, the only exception being that when an existing entry is being edited, the previously saved parameters appear in the different dialog box sections of the Wizard.

To define a DC-DC converter model:

1. In the DC-DC Converter Model Selection, select a model from the list, or click **New** to add a new model.

   The Library Builder classifies all DC-DC converters and regulators into four different families:

   - Supplies that can be adjusted up or down a few percent about a nominal voltage by connecting a resistor between the supply's TRIM pin and its output or ground.
   - Supplies that can be adjusted over a wide range by connecting a resistor between the supply's TRIM pin and ground.
   - Supplies that can be adjusted over a wide range by connecting a resistor between the supply’s TRIM pin and Vout.
   - Supplies that use an external feedback network. The Library Builder calls this a “discrete” implementation.

2. Click **Next** to open the Select the type of DC-DC Converter section.
3. Select the type of DC-DC Converter and click **Next**.

   The wizard opens the configuration section of the dialog box, which enables you to set the parameters for the type of converter you selected.

   For the Trim-up Trim-down supply type and the Programmable supply type, the dialog box asks for several trimming examples to be entered, and the supply's adjustment characteristics are determined from the data. Many power supply data sheets provide tables of resistor values needed to achieve various different outputs. These examples can be entered directly into the dialog box.

   For discrete implementation types, the dialog box requests the values of the feedback network components and the internal reference voltage. From the data, the supply's adjustment characteristics are determined. When a specific model of voltage regulator will be used several times to produce different voltages, separate library entries will be needed for each unique set of feedback network components.

4. After setting the configuration parameters, click **Finish**.

See Also “Setting Multiple VMON/Trim Properties” on page 127

“Setting VMON Properties” on page 128

“Setting Trim and Margin Properties” on page 129

Configuring Voltage Identification (VID) Tables

Voltage identification tables define the target voltage outputs of a DC-DC converter based on inputs of a selected bus. The VID section of the Voltage Monitor and Control dialog box enables you to select and configure a VID table for the selected voltage monitor. If your design does not yet contain VID tables, you can build them and add them to the VID Table Library.

Building VID Tables The Voltage – VID Table Library interface enables you to build a library of voltage identification tables. By default, Platform Designer stores the VID tables inside the Diamond installation sub-directories. You can specify a different location in the Global ASC Options section of the Global Configuration View.

To create a new VID table:

1. In the Voltage editor, open the VID Table library dialog box by using one of the following methods:
   - Open the Voltage Monitor & Control Properties dialog box by double-clicking the VMON/TRIM cell. Select the VID tab, and choose **Add / Edit a table** from the VID Lookup Table drop-down menu.
   - In the Global Configuration View, select the ASC Options tab. In the VID Options section on the right, click **Add / Edit a table**.
2. In the VID Table Library dialog box, click **New** to open the Create New VID Table dialog box.
3. Type a name for the VID table in the text box, and then select the table size from the drop-down menu.
4. In the Auto fill section, specify a voltage starting value and step size.

   Alternatively, you can manually enter each value in the table by double-clicking each Voltage (V) cell and typing a value. If you choose manual entry, skip Step 5.
5. Select either the Top Down or Bottom Up option, and then click **Fill**.

   Platform Designer fills in the voltage for each VID value, based on the step size and starting value. You can edit any of these values as desired.
6. Click **OK**.

   The new VID table is added to the VID Library and will be listed in the VID Table menu in the Voltage editor and in the Voltage Monitor & Control Properties dialog box.

Selecting and Configuring the VID Table The VID page of the Voltage editor enables you to select the VID lookup table from the VID Table Library and select the bus, strobe port or node, and VID strobe edge. You can also select and configure the VID table from the Voltage Monitor & Control dialog box.

Editing a VID Table To edit a VID Table, open the VID Table library dialog box by using any of the methods described in Step 1. Select a VID table and

click **Edit**. In the Edit the VID Table dialog box, change the settings as desired and click **OK**.

See Also "Configuring Voltage Sense (VMON) and Control (TRIM)" on page 126

Configuring Current Sense (IMON)

The Current editor allows you to set parameters for the current monitor (IMON) and high-voltage current monitor (HIMON) for each Analog Sense and Control device. Each monitor includes settings for two precision comparators (A and B) and one fast (F) comparator.

- Schematic Net Name – Each IMON or HIMON must be identified by a unique name. The name can consist of any combination of alphanumeric characters.
- Logical Name – The monitor's comparator outputs A, B, and F must each be identified by a unique name. The name can consist of any combination of alphanumeric characters.
- Sense Resistor (Ohm) – The sense resistor value will be the same for all comparators within the monitor and can range from .0001 to 1000000.0 ohms.
- Trip Point Selection (Amps) – The selected trip point, for comparators A, B, and F, will be associated with each logical name. The trip points available are determined by the Sense Resistor value.
- Hysteresis – Hysteresis can be enabled or disabled for comparators A and B. This automatically updates the available trip point selections.
- Glitch Filter (µ) – The glitch filter can be enabled or bypassed for comparators A and B. When Yes is selected, this option configures the comparators so that supply glitches narrower than 64 microseconds are ignored.
- Window Mode – Window mode can be used for comparators A and B. To use the window mode, the Comparator B threshold should be lower than the threshold setting of comparator A. The window mode output will replace the comparator A output. The window output is logical high if the Comparator B output is high and the Comparator A output is Low.
- Low Side Sense – When Yes is selected, this sets the low voltage side sense for comparators A, B, and F. It is not available for high-voltage sense monitors.

The Programmable Gain Amplifier (PGA) Gain setting is displayed by the tool based on the selected trip points. This information is used when performing A/D measurements of the current over I^2C.

The V-Drop (V) and Peak Power (W) are calculated based on the values entered for trip point and sense resistor.

See Also "Configuring Voltage Sense (VMON) and Control (TRIM)" on page 126

Configuring Temperature Sense (TMON)

The Temperature editor allows you to set parameters for the external temperature sensor interfaces (TMON1, TMON2, etc) and the internal temperature sensor (TMON_Int). The temperature monitor function includes an A/D converter measurement with averaging support, as well as an alarm monitor function with programmable hysteresis and filtering. Each TMON channel includes two separate programmable comparison points. The temperature sensor hardware also supports a configurable offset and ideality factor. The parameters for each temperature monitor comparator can be set in Platform Designer's Temperature editor, as follows:

- Schematic Net Name – The schematic net name must be unique and can consist of any combination of alphanumeric characters.
- Logical Name – This must be a unique name that will identify each TMON pin and comparator output. It can consist of any combination of alphanumeric characters.
- Monitoring Type – Over temperature (OT) or under temperature (UT) monitoring can be selected.
- Trip Point Selection – This sets the trip point for the comparator in degrees Celsius.
- Hysteresis – This defines the hysteresis for updating the trip point output for the comparator.
- Monitor Alarm Filter (Depth) – This sets the number of consecutive alarms that must be detected before setting the comparator output high.
- Measurement Averaging (Filter Coefficient) – This sets the coefficient on the averaging filter implemented on the temperature sensing of the TMON.
- Offset – This calibration value is used to correct the temperature reading, in degrees Celsius for the TMON.
- Short Fault Measurement Reading – This defines the measurement reading under a short fault condition. It implies the Open fault measurement reading as well. See the Platform Manager 2, MachXO2HC/HE, MachXO3D, or MachXO3LF data sheet for more details.
- Ideality Factor – This calibration value is used to correct the diode imperfection for the TMON.
- Sensor Configuration – The external temperature monitors can be configured to interface with several different transistor circuits.

See Also "Configuring Fan Control" on page 135

"Configuring Voltage Sense (VMON) and Control (TRIM)" on page 126

Configuring Fan Control

Platform Designer's Fan Controller provides fan control for 2-wire, 3-wire, or 4-wire fans. The fan controller supports multiple fan speeds, under-speed alarm detection, and a configurable startup pulse. It enables you to set three levels of speed control through the pulse width modulated (PWM) signal. Up to 18 separate fans can be controlled.

To configure fan control:

1. Select **Fan Controller** from the menu pane, and then select a fan in the "Fan in System" list or add a new fan by clicking the add button  on the lower left.
2. In the Type Setting section on the Parameters page, select the fan type from the drop-down menu.
3. If you selected 3 Wire (Low Side Drive), 3 Wire (High Side Drive), or 4 Wire, also specify the **Sense Type** for alarm detection, as follows:
   - If the fan supports a locked signal as fan feedback to indicate that the fan is locked or stuck, select **Alarm (Active High)** or **Alarm (Active Low)**.
   - If the fan includes a tachometer feedback signal, select **Tach** (2 Pulses/Rev), and then do the following:
     - In the Alarm Interval box, type a value for the number of ms that will be allowed between pulses before an under-speed violation is registered.
     - Select a value between 1 and 10 from the Alarm Filter Count. This is the number of times the alarm interval must be detected before the Alarm signal will be asserted.

     With tachometer feedback, a +/- 10% hysteresis will be applied to the corresponding assertion/de-assertion of the failure count to prevent a false alarm.
4. In the PWM Setting section, use the sliders to select a duty cycle percentage for Speed3, Speed2, and Speed1.

   Each can be set in 5% increments, ranging from 5% to 95%. The fastest speed must be assigned to Speed3, with the middle speed assigned to Speed2, and the slowest speed assigned to Speed1.
5. Use the slider to specify the PWM frequency. The frequency can be set in increments of 0.8 kHz, up to 80.0 kHz.
6. Select the polarity for the PWM signal, based on the nature of the MOSFET switch and on-board buffers.

   Refer to the fan data sheet and the board requirements for information about the correct setting of PWM polarity.
7. Optionally, enable the Startup Control and select the pulse duration, in increments of .25s for the startup period.

   When startup control is enabled, the fan controller will send a kickstart pulse of the chosen duration to the fan before switching to the selected PWM setting.

8. Select the InterConnections tab.

   The InterConnections page allows you to specify the signals that the Platform Designer will use as fan controller ports. Each fan has an independent set of interconnections.

   The PWM Out and Tach/Alarm Sense are the external connections to the fan. The other signals (Speed Sel 1 ... Alarm Out) are logical connections that set the operation of the fan controller during runtime.

9. Assign the PWM Out and Tach/Alarm Sense to external ports by dragging the desired ports from the Signal Pool.

10. Drag the desired signals from the Signal Pool to set the logical connections for Speed Sel, Run Off, Run Full, and Alarm Out.

See Also ▶"Configuring Temperature Sense (TMON)" on page 134

▶ "Configuring Voltage Sense (VMON) and Control (TRIM)" on page 126

Setting Fault Logging Options

The Fault Logger, available from the Components section of the menu pane, enables you to specify and configure fault conditions that are logged to non-volatile memory. You have the option of using standard fault logging that is based on voltage, current, and temperature status; or using the full-featured fault logging. By default, fault logging is disabled.

Each ASC device contains a block of EEPROM memory that can be used for either the V, I, T Fault Log or for User Tag Operation. The User Tag Memory is enabled automatically when fault logging is disabled or when full-featured fault logging is selected. It is disabled when V, I, T fault logging is selected.

V, I, T Fault Logging When you select this option, the Analog Sense and Control device will log the voltage (V), current (I), and temperature (T) status when a fault is detected. The V, I, T fault log supports the logging of 8 signals per Analog Sense and Control device, in addition to the V, I, T signals. These can be digital I/O or other signals.

To configure V, I, T fault logging:

1. In the Fault Logger, select **Enable V, I, T Fault Log** from the options shown at the top left.
2. Select a signal name from the Fault Logging Trigger Signal Name drop-down menu. This signal can be an output from the user logic, an external digital I/O, or a specific monitor alarm signal.
3. In the "Basic Configuration" section, select the type of memory to log to: EEPROM or SRAM. The EEPROM is non-volatile, but a log takes longer to execute. The SRAM memory is volatile but is executed more quickly.
4. The V, I, T fault log supports the logging of 8 signals per Analog Sense and Control device in addition to the V, I, T signals. These can be digital I/O or other signals. For each signal that you want to log, drag it from the "User Signals to Log" list over to a D cell of the desired ASC.

Full-Featured Fault Logging The full-featured fault logging uses either the UFM Flash memory within the device or an external SPI flash memory. Full-featured fault logging is available for Platform Manager 2 LPTM21 devices and for MachXO2HC/HE devices of 1200 and above, MachXO3D, and MachXO3LF devices. ECP5U/UM device support full-featured fault logging with external SPI flash memory only - UFM is not available in ECP5U/UM.

To configure full-featured fault logging (Platform Manager 2, MachXO2HC/HE, MachXO3D, and MachXO3LF only):

1. In the Fault Logger, select **Enable Full-Featured Fault Log** from the options shown at the top left.
2. Select a signal name from the Fault Logging Trigger Signal Name drop-down menu. This signal can be an output from the user logic, an external digital I/O, or a specific monitor alarm signal.
3. n the “Basic Configuration” section, do the following:
    a. From the drop-down menu, select the contiguous ASCs that you want logged.
    b. Select **Include Timestamp** if you want Platform Designer to include a 32-bit timestamp timer.
    c. Select a signal to be used as the busy signal name from the “Busy” Signal Name menu. This signal is an output from the fault logger and can be used to inform the user logic that a fault log recording is in progress.
    d. Select a “Log to” option: the on-chip UFM memory or external SPI flash.
    e. If you selected External SPI Flash as the “Log to” option, use the Model drop-down menu to select the external flash model to be used for fault logging.
4. In the User Log Fields menu, located above the user log table on the right, select the number of additional bits to record in the fault log.
5. For each additional signal that you want to log, drag it from the “User Signals to Log” list over to a D cell of the desired user log column.

To configure full-featured fault logging (ECP5U/UM only):

1. In the Fault Logger, select **Enable Full-Featured Fault Log** from the options shown at the top left.
2. Select a signal name from the Fault Logging Trigger Signal Name drop-down menu. This signal can be an output from the user logic, an external digital I/O, or a specific monitor alarm signal.

    Select a signal to be used as the busy signal name from the “Busy” Signal Name menu. This signal is an output from the fault logger and can be used to inform the user logic that a fault log recording is in progress.
3. In the “Basic Configuration” section, do the following:
    a. From the drop-down menu, select the contiguous ASCs that you want logged.

b. Select either Disable Time Stamp, Internally Generated Time Stamp, or Externally Assigned Time Stamp. The internally generated timestamp option includes the 1s per bit, 32-bit timestamp. The externally assigned timestamp option will automatically update the Nodes view to include a 32-bit node group, called FL_Timestamp, which can be assigned by the user based on an external, highly accurate timestamp.

c. In SPI Memory Configuration, select Log to Configuration SPI Memory or Log to Alternate SPI Memory.

d. Use the Model drop-down menu to select the external flash model to be used for fault logging.

4. In the User Log Fields menu, located above the user log table on the right, select the number of additional bits to record in the fault log.
5. For each additional signal that you want to log, drag it from the “User Signals to Log” list over to a D cell of the desired user log column.
6. In the “Fault Log Memory Space” section, do the following:

   a. Enter the Start Address. This is a Hex input used by the IP to determine the starting address available to the fault logger. This setting allows the user to share the SPI memory with multiple boot images or other information. The default address is 0x030000.

   b. Enter the size in K Bytes.

   c. The End Address is automatically calculated and displayed by Platform Designer. The displayed value confirms for the user that it is safe to share the SPI memory with multiple boot images or other information.

7. The full-featured fault logger automatically adds a 16-bit node group called FL_Record_Counter to the nodes view. This nodes group can be connected to user logic to provide the number of Fault Records currently stored in memory.

See Also ▶“Configuring a SPI Flash Model” on page 125

Configuring Hot Swap

(Platform Manager 2. MachXO2HC/HE, MachXO3D, and MachXO3LF only.) Platform Designer's Hot Swap utility enables you to configure a hot swap component that will control the in-rush current in applications where power is rapidly applied, such as a circuit board that is plugged into a backplane. The Hot Swap utility contains a separate page for each of the following three tasks:

- Hot Swap – Specify the settings of a hot swap component.
- Interface – Connect the hot swap component to the Platform Manager 2 resources.
- InterConnections – Connect the hot swap component to the hardware management logic.

To specify the settings of a hot swap component:

1. Select **Hot Swap** from the Components section of the menu pane.
2. On the Hot Swap page, select a hot swap component from the list or create a new hot swap component by clicking the add button  on the lower left.
3. Specify the desired options and settings:
   - Input Supply
     - V_{IN} – Input voltage. This is the supply voltage input, which is used to charge C_{LOAD}. Together with I_{MAX}, C_{LOAD} and the MOSFET SOA (Safe Operating Area) information will be used to calculate the hot swap parameters, including the number of phases, voltage and current trip points.
     - I_{MAX} – Maximum current available from the input supply.
   - MOSFET Parameters
     - Load/Supply – Allows you to place the MOSFET at either the Supply or the Load. Select the setting that matches your application.
     - MOSFET Name – Used, for informational purposes only, to track the part number of the MOSFET in your application.
     - V1/A1 – Voltage/Current data point pair from the MOSFET SOA curve found in the MOSFET datasheet.
     - V2/A2 – Voltage/Current data point pair from the MOSFET SOA curve. V1 must be greater than V2, and A1 must be less than A2.
   - Current Sensing Configuration
     - R_S – Sense Resistor used in the application. Platform Designer uses this sense resistor value to calculate available current sensing trip points.
     - Current Sense Amp choices – Determines the current sense amp circuit used in the application.

       The choice among “External and VMON,” “External and IMON,” and “IMON (Internal)” will determine whether additional current sensing circuit information is required. The choice also determines which monitor circuits are available for assignment in the Interface tab. Certain configurations of V_{IN} and Current Sense Amp are prohibited, such as using IMON (Internal) with V_{IN} voltages above 13.2V.
     - Low Side option – Determines whether the IMON is used in a near ground hot swap application (such as -48V). Selecting this option enables the Low-Side Sense feature in the IMON circuit.
   - Hot Swap Settings
     - C_{LOAD} – The load capacitor charged up during the hot swap operation. This value is used to calculate the hot swap parameters, including the number of phases and the voltage and current trip points.

- Fast Shutdown Limit – Sets the current limit for fast shutdown during hot swap. The menu is automatically populated with available current limit trip points related to the Sense Resistor and Current Sense Amp settings. If the current exceeds the selected limit during hot swap, the hot swap will be aborted and the over current error signal will be asserted. This feature can be disabled by selecting “None.”
- Cooldown Period – The period used by the hot swap algorithm to ensure that the MOSFET does not overheat. Each time the hot swap algorithm detects a timeout, remaining in a phase longer than 10 ms, the cooldown period will be used as a wait time prior to resuming the hot swap operation.
- Hot Swap Name – A user-defined name for the hot swap setup.

4. After all the parameters have been selected, click the **Calculate** button.

   The V_{LOAD} and I_D graphs are updated with the estimated Voltage and current profiles based on the calculated Hot Swap algorithm and settings. The Results/Comments area is updated with information, including the number of hot swap phases and the number of VMON and IMON channels required to support the hot swap implementation.

   The hot swap calculation might not be able to provide a solution for all circuit configurations. In this case, error information is displayed in the Diamond message console.

5. If the results are not satisfactory, adjust the settings and calculate again.

To connect the hot swap component to Platform Manager 2 resources:

1. After you are satisfied with the hot swap calculation results, select the **Interface** tab.

   The Interface page consists of several menus that are used to assign VMONs, IMONs, HVOUT, GPIO and PIO channels to the different hot swap functions. When you assign VMON, IMON, and HVOUT channels in the Interface tab, their settings, such as trip point thresholds, will be updated automatically.

2. Select the desired setting from each of the drop-down menus in the gold-colored boxes, as follows:

   - V_{IN} Monitor – Associates a VMON channel with the input voltage monitoring function of a given hot swap. This VMON channel can be re-used across multiple hot swap components, if they use the same input voltage. The V_{IN} Monitor can be from any ASC in the Platform Manager 2 system.

     Selecting a V_{IN} Monitor is mandatory.

   - ASC – Specifies the ASC device to be used. The ASC menu is automatically populated with the ASCs used in the system. All channels from the drop-down menus inside the pink rectangle will be selected from the chosen ASC.

- MOSFET/Charge Pump Driver – Associates an HVOUT or GPIO with the controlled output signal used in the hot swap algorithm.

  Selecting an HVOUT or GPIO is mandatory.

- Current Sensing Monitor – Specifies the IMON or VMON channel for current sensing. The channels available are those that support the configurations that were set on the Hot Swap page.

  Selecting a current sensing monitor is mandatory.

- Fast Current Sense Monitor – Specifies an IMON or VMON channel for fast current sense. This menu is used only when a fast shutdown limit is selected on the Hot Swap page. It is used in conjunction with the fast current sense output, fast path input, and fast shutdown signal.

- Fast Current Sense Output – Specifies an HVOUT or GPIO as a fast output signal. This menu is populated with the output signals available on the ASC.

- Fast Path Input – Specifies a signal for fast path input. This menu is populated with the PIO signals from the FPGA I/O used in the Platform Manager 2 system. The signal should be externally connected to the fast current sense output signal in the application.

- Fast Shut Down Output – Specifies a signal for fast shutdown output. This menu is populated with the PIO signals from the FPGA I/O used in the Platform Manager 2 system. The signal is controlled by the hot swap IP and will perform a fast shutdown of the MOSFET when necessary.

- Load Voltage Monitor(s) – Specifies the load VMON or HVMON. This menu is populated with the voltage monitor channels from all ASCs in the Platform Manager 2 system. The number of drop-down menus available is determined by the number of phases required by the hot swap algorithm. These menus can be filled with voltage monitor channels from any ASC in the system.

  Selecting a load voltage monitor from each available menu is mandatory.

To connect the hot swap component to the hardware management logic:

1. Select the **Interconnections** tab.

   The Interconnections page includes a set of control signals that are used to connect the hot swap component to the hardware management logic. Each of these signals is typically assigned to a node.

2. Drag the desired signal from the Signal Pool over to the appropriate box, as follows:

   - Hot Swap Enable – Input signal to the hot swap component. Asserting this signal high will start the hot swap operation.

   - Over-Current Clear – Input signal to the hot swap component. This signal needs to be asserted to clear any existing Over-Current Error status.

- Hot Swap Done – Output signal from the hot swap component. This signal is asserted by the hot swap component to inform the hardware management logic that the hot swap operation was successfully completed. This signal is cleared when the hot swap enable signal is asserted again.
- Hot Swap Error – Output signal from the hot swap component. This signal is asserted by the hot swap component to inform the hardware management logic that an error occurred during the hot swap operation. This signal is cleared when the hot swap enable signal is asserted again.
- Over-Current Error – Output signal from the hot swap component. This signal is asserted by the hot swap component to inform the hardware management logic that an overcurrent error occurred during the hot swap operation. This is in addition to the fast path shutdown behavior. This signal will be cleared when the over-current clear signal is asserted.

Configuring PMBus Adapter

(Platform Manager 2, MachXO2HC/HE, MachXO3D, and MachXO3LF only.) PMBus is a serial communication bus that allows a microcontroller to configure and monitor Digital Point of Load (DPOL) DC-DC Converters. The PMBus Adapter allows the connection of analog POLs to the PMBus controller using the FPGA and ASC. Multiple POLs may be mapped to different pages configured in the PMBUS adapter. Dedicated pages for voltage, current, and temperature monitoring are provided.

The PMBus adapter supports Packet Error Checking and SMBAlert options. The PMBus adapter supports a pre-defined subset of PMBus commands.

To configure PMBus Adapter:

- In the PMBus Adapter tab, select **Enable PMBus Adapter** at the top left.

  The PMBus tab contains controls and the ASC Page Mapping table control. When **Enable PMBus Adapter** is checked, controls and the related PMBus Nodes tab are enabled. To view PMBus ports and nodes, refer to “Viewing PMBus Ports and Nodes (Platform Manager 2, MachXO2HC/HE, MachXO3D, and MachXO3LF Only)” on page 144.

The following options are available to configure PMBus Adapter:

Enable packet Error Checking (PEC) Enables/disables the packet error checking option.

Enable SMBAlert Enables/disables the SMBAlert output pin..

PMBus Bus Speed Specifies PMBus bus speed. The two available values are 100KHz and 400KHz.

PMBus Slave Address Specifies PMBus slave address. Only legal slave addresses are selectable from the drop-down list.

ASC Page Mapping Specifies the PMBus page mapping of voltage monitors (VMONs), Current Monitors (IMONs) and Temperature Monitors (TMONs) of the ASC devices in the design. The resource list view on the left contains the VMON , IMONs, and TMON signals that are part of the current design. The user can drag and drop the VMON. IMON, and TMON signals from the resource list view and assign them to PMBus pages on the right.

VMON, IMON, and TMON signals can be assigned to page numbers as follows:

- VMON - 0x00-0x2F
- IMON - 0x30-0x3F
- TMON - 0x40-0x5F

The default values on the ASC page mapping table views is None. When the user drags the VMON, IMON, or TMON signals from the resource list on the left and drops it in the table view, a dynamic design rule check (DRC) is performed in the background to verify whether the assigned page is valid or conflicts with already assigned pages.

Implementing the Platform Management Algorithm

Implementing the platform management algorithm involves labeling and configuring ports and nodes, and then building logic instructions based on these configurations. In Platform Designer, it involves the use of the Ports & Nodes editor and the Logic editor.

See Also ▶"Designing Control Sequences" on page 150

Configuring and Viewing Ports and Nodes

Use the Ports & Nodes editor to view internal nodes and configure external signals (ports) of the Platform Manager 2, MachXO3D, MachXO2HC/HE, MachXO3LF, ECP5U/UM device; configure and label the ASC-based general-purpose I/Os; view PMBus Ports and Nodes; and label and configure the high-voltage output ports. You can build control logic based on the labels you apply.

See Also ▶"Designing Control Sequences" on page 150

Configuring Ports

The Ports section of the Ports & Nodes editor shows all physical FPGA-based PIOs that are available for use as external connections to the design.

You can assign a user label to a logical port. A port requires a type: IN, OUT, INOUT or GRESET. The port also includes attributes for group, register type and reset level.

For register type, in addition to Registered or Combinatorial, you have the option of using “Registered Sync with ASC.” The MachXO2HC/HE, MachXO3D, MachXO3LF, ECP5U/UM device output is much faster than the ASC output because of the 3-wire interface propagation delay. So when you select the “Registered Sync with ASC” option, Platform Designer will add a delay for MachXO2HC/HE outputs to make sure that they are triggered in the same step.

After the design is compiled, you can cross-probe from the Ports sheet to Diamond’s Spreadsheet View or Netlist View to make pin assignments. Right-click a cell and choose **Show In > Spreadsheet View** or **Show In > Netlist View**.

Viewing PMBus Ports and Nodes (Platform Manager 2, MachXO2HC/HE, MachXO3D, and MachXO3LF Only)

The PMBus tab is enabled only if the PMBus Adapter is enabled. For information on enabling PMBus Adapter, refer to “Configuring PMBus Adapter” on page 142.

The PMbus tab lists all of the available PMBus ports/nodes that can be connected to the Logic Controls, including Sequence, Supervisory equation, and HDL Import. These nodes are managed in the user logic in order to complete the configurable PMBUS support.

The PMBus ports/nodes table view includes three read-only columns:

Logical Name/Function The node names listed in this column are the nodes that are added to the user signal pool and which connect the logic to the PMBus Adapter for support of PMBus commands..

Grouped By This column lists the PMBus status word or byte name with the correct bit index. Nodes that are not used are set to zero.

Logic Connection This column shows the node direction with respect to the user logic; In goes into the user logic and Out is driven from the user logic.

See Also ▶“Creating a Port Group” on page 144

- “Designing Control Sequences” on page 150
- “Working with Platform Designer Editors” on page 120

Creating a Port Group

The group feature allows you to combine ports of a common type, using vector notation, to form a bus. The selected ports cannot be shared among multiple groups.

To create a port group:

1. On the Ports sheet or the Nodes sheet, right-click any cell of a port or node that is not already part of a group and choose **Add Group** from the pop-up menu.
2. In the New Group dialog box, type a name for the group in the Group Name text box.
3. From the Available Ports list, select the ports that you want to include in the group, and then click the > button to place the ports in the Selected Ports list.
4. Click **Add**.

To modify a port group:

1. Right-click a cell in the Group By column that references the port group you want to edit and choose **Modify Group** from the pop-up menu.
2. In the Modify Group dialog box, use the appropriate buttons to add ports to the Selected Ports list or remove them from the list.
3. Click **Update**.

To remove a port group:

- Right-click any cell and choose **Remove Group**. From the sub-menu, select the group you want to remove.

See Also ▶"Designing Control Sequences" on page 150

- "Working with Platform Designer Editors" on page 120

Configuring Nodes

The Nodes sheet enables you to add and configure internal nodes in your design. You can assign user labels to new and existing nodes and select the register type and reset level attributes.

To add a node:

1. Press the Insert key, or right click an existing node and choose **ADD NODE**.
2. Assign a unique name to the node and select the Register Type and Reset Level.

To delete one or more nodes:

- Highlight the rows you want to delete by selecting the row numbers in the first column, and then press the Delete key. Alternatively, right-click and choose **REMOVE NODE(s)**.

Nodes can also be grouped, following the same process as described in "Creating a Port Group" on page 144.

Some nodes are automatically allocated to the component logic or other features such as fault log enable and timers.

See Also "Designing Control Sequences" on page 150

"Working with Platform Designer Editors" on page 120

Configuring General-Purpose I/Os

The GPIO sheet allows you to enable and assign a user label to an ASC-based general purpose I/O.

For each GPIO pin, you can specify the following:

- A unique name to identify the pin.
- The type: IN or OUT
- Reset Level: Set High or Set Low

For pins GPIO2 and GPIO3 of each ASC, you can also select the following:

- Source
- Polarity: True or Inverted

See Also "Designing Control Sequences" on page 150

Configuring High-Voltage Outputs

The HV Outputs sheet allows you to enable and assign a user label to a high-voltage output port.

For each HVOUT pin, you can specify the following:

- A unique logical name to identify the HVOUT pin.
- The output setting: Charge Pump or Open Drain
- The target voltage for the output (Charge Pump mode only)
- The source current in micro-amps for the output (Charge Pump mode only)
- The sink current in micro-amps for the output (Charge Pump mode only)
- The output mode: Static or Switched
- The selected frequency for the ouput in kHz (Switched mode only)
- The duty cycle for the output (Switched output mode only)
- The output source
- The output polarity: True or Inverted
- A high or low reset level

Designing with the Logic Editor

Included with the Lattice Diamond Platform Designer is the LogiBuilder™ Logic editor, which enables you to define a power supply sequence controller and monitor or other control circuits for implementing the platform management algorithm. The tools include a set of instructions for building the sequence based on conditional events and timer delays.

See Also

- "Designing Control Sequences" on page 150
- "Sequence Controller Instruction Set" on page 148

About the Logic Editor

The Logic editor simplifies the design process by allowing you to select from menus instead of writing complex code. It enables you to add and configure timers and import HDL modules. It also supports vector notation for bus structures.

After entering the set of instructions, you can compile the design and simulate the sequence or control events. When the design is compiled, synthesizable HDL code is generated based on the sequencer and supervisory logic.

- "Designing Control Sequences" on page 150
- "Sequence Controller Instruction Set" on page 148

Logic Editor User Interface

The LogiBuilder Logic editor includes separate views for building the logic instructions and exceptions, entering supervisory equations, and defining timers. An additional view is provided that allows you to import HDL modules into your design.

- Sequence – The Sequence view enables you to define the step-by-step instructions for controlling outputs for a selected state machine. When compiled, sequencer instructions implement a digital logic state machine within the PLD core.
  - Exceptions – The Exceptions section allows you to define equations that will trigger sequence controller exceptions to modify outputs and jump out to an alternative sequence step. Exceptions can be selectively applied to any sequencer step. When compiled, exception instructions are merged with the digital logic state machine of the PLD core.
  - State Machine – A tab for each state machine is shown at the bottom of the Sequence view, which allows you to select the active state machine for editing. You can also add a new state machine by clicking the **+** tab. Right-click a state machine tab to open the Multiple State Machine dialog box, which enables you to add and delete state machines.

- Supervisory – The Supervisory view enables you to define combinatorial and registered logic independent of the sequencer control logic. When compiled, supervisory equations are concurrent to the digital logic state machine of the PLD core.
- Timers – The Timers view enables you to define timers for control sequences. To add a new timer, press the Insert key or click the **Add** button.
- Imported HDL – The Imported HDL view enables you to add HDL modules from HDL source that is included in the design project.

See Also
- "Designing Control Sequences" on page 150
- "Managing Multiple Control Sequences" on page 151.
- "Entering Supervisory Equations" on page 155
- "Defining Timers" on page 151
- "Sequence Controller Instruction Set" on page 148

Sequence Controller Instruction Set

The LogiBuilder Logic editor of Platform Designer provides the following instructions for designing control sequences:

BEGIN STARTUP SEQUENCE The BEGIN STARTUP SEQUENCE instruction signals to LogiBuilder that any instructions past this point may be interrupted by jumps specified in exceptions. This instruction may be deleted from a sequence, but not inserted.

OUTPUT The OUTPUT instruction is used to turn on or turn off the output signals. A single OUTPUT instruction can be used to simultaneously change the status of any number of output signals.

WAIT FOR <Boolean Expression> The WAIT FOR <Boolean expression> instruction suspends execution of the sequence until the specified expression becomes TRUE. Outputs can be assigned in this instruction. These outputs are asserted as soon as the sequencer enters the instruction.

WAIT FOR <Boolean Expression> with Timeout The WAIT FOR <Boolean expression> with Timeout instruction suspends execution of the sequence until the specified expression becomes TRUE or the selected timer expires. The timer is started when the sequence enters the instruction.

WAIT FOR <timeout> The WAIT FOR <timeout> instruction is used to specify a fixed delay in the execution sequence. The value of <timeout> is determined by which timer is specified. (The timer must first be configured on the Timers page).

IF <Boolean Expression> THEN GOTO <step x> ELSE GOTO <step y> The If/Then/Else instruction provides the ability to modify sequence flow depending on the state of inputs. If <Boolean expression> is TRUE, the next step in the sequence will be <step x>, otherwise the next step will be <step y>.

IF <timeout> THEN GOTO <step x> ELSE If <Boolean Expression> GOTO <step y> ELSE GOTO <step z> This instruction provides the ability to modify sequence flow depending on the state of inputs with an additional timeout feature. If Timer <n> has expired, the next step in the sequence will be <step x>; otherwise, if <Boolean expression> is TRUE, the next step will be <step y>. If <Boolean expression> is FALSE and Timer <n> has not expired, then the next step will be <step z>. This instruction only checks the values of <Boolean expression> and Timer <n>; it does not start or reset the timer.

GOTO <step x> The GOTO instruction forces the sequence to jump to <step x>.

Start Timer This instruction starts the selected timer. The status of the timer must be checked using another instruction or combinational logic.

Stop Timer This instruction stops and resets the selected timer.

NOP The NOP instruction does not affect any of the outputs or the sequence of execution. It is effectively a single-cycle delay.

HALT The HALT instruction stops execution of the sequence.

BEGIN SHUTDOWN SEQUENCE The BEGIN SHUTDOWN SEQUENCE instruction signals to LogiBuilder that any instructions past this point will not be interrupted by jumps specified in exceptions. This feature allows code used for handling exceptions not to be interfered with by other exceptions that may occur. This instruction may be deleted from a sequence, but not inserted.

General Information on Instructions

All instructions are able to assign outputs, except for NOP, BEGIN STARTUP SEQUENCE, and BEGIN SHUTDOWN SEQUENCE. The outputs are assigned immediately when the sequencer enters the instruction.

The GOTO and If/Then/Else instructions cannot jump to the Wait for (timeout value), Wait for (Boolean) with Timeout, or Start Timer instruction.

See Also

- "Designing Control Sequences" on page 150
- "Managing Multiple Control Sequences" on page 151
- "Editing Sequence Instructions" on page 152
- "Entering Supervisory Equations" on page 155

Designing Control Sequences

The sequencer instructions define the steps for controlling selected outputs. The exceptions define the equations that will trigger sequence controller exceptions to modify outputs and jump out to an alternative sequence step. Exceptions can be selectively applied to any sequencer step. When compiled, the control sequence instructions implement a digital logic state machine within the PLD core.

To design a control sequence:

1. In the LogiBuilder Logic editor, select the **Sequence** tab. If your design uses multiple state machines, select the one that you want to use for the sequencer instructions.
2. In the sequence (upper) portion of the editor, click **Step 1 Begin Shutdown Sequence** to highlight it.
3. Double-click Step 1 or press the **Insert** key on your keyboard to open the Insert Step Dialog box.
4. In the dialog box, choose an instruction type, and click **OK**. Repeat as necessary to add sequence steps.
5. Double-click each instruction step to open the appropriate Edit Properties dialog box.
6. Select the desired instruction properties in the Edit dialog box, and click **OK**.
7. To add exceptions, do the following:
   a. Double-click ***<end-of-exception-table>*** in the exceptions (lower) portion of the editor, or highlight it and press the **Insert** key.
   b. Repeat as necessary to add additional exceptions.
   c. Double-click each exception placeholder, to open the Exception Properties dialog box. Alternatively, right-click the placeholder and choose **Properties**.
   d. Click the **Edit** button at the top right of the dialog box to open the Boolean Expression Editor. Set the expression that will trigger the exception and click **OK**.
   e. In the Exception Properties dialog box, select the desired exception properties, and click **OK**.

See Also

- "Sequence Controller Instruction Set" on page 148
- "Editing Sequence Instructions" on page 152
- "Editing Exceptions" on page 155
- "Managing Multiple Control Sequences" on page 151

Managing Multiple Control Sequences

The LogiBuilder Logic editor supports multiple state machines for power-up sequence and control. The state machines are defined separately but can interact through nodes or common logic functions. Each state machine is built up in a separate tab in the Sequence section of the Logic editor.

To manage multiple control sequences:

1. In the LogiBuilder Logic editor, select the Sequence tab.

   The name of each currently defined state machine is displayed in a separate tab at the bottom. You can add a new state machine by clicking the tab that contains the **+** sign. To delete a state machine, you must use the Multiple State Machines dialog box.

2. Right-click a tab for a state machine and choose **Multiple State Machine** from the pop-up menu. Alternatively, use the **Ctrl+M** keyboard shortcut.

3. In the Multiple State Machines dialog box, do one or all of the following:

   - Select a state machine and type a unique name for it in the State Machine Name text box.
   - Delete a state machine by selecting it and clicking **Delete SM**.
   - Click **Add SM** to add a new state machine.

4. Click **OK**.

See Also ▶"Sequence Controller Instruction Set" on page 148

- "Designing Control Sequences" on page 150

Defining Timers

Use the LogiBuilder Logic editor to set up timers for control sequences. After you have defined timers, they will be displayed in the properties dialog boxes for sequences that include timeouts. This will allow you to select the timeout value from the list.

To define a timer:

1. In the Logic editor, select the Timers tab.
2. Click the **Add** button at the bottom or press the **Insert** key to create a new timer.

   The timer is added to the list and given a default name.

3. If desired, click the default name in the Timer Name column and type a unique name.
4. Click the Clock Source cell and select a source from the drop-down menu. ECP5U/UM only support the 62.5 kHz logic sequence clock as a timer source clock.

5. Click the Period Cell. Select a time unit from the drop-down menu, and then specify the value.

   Note

   Timer delay settings must be longer than 160 µs.

   The number of LUT resources required to generate the timer is automatically recalculated.

See Also ▶"Sequence Controller Instruction Set" on page 148

Editing Sequence Instructions

The dialog boxes provided by the Logic editor enable you to set up and edit sequence instruction properties, Boolean expressions, and exceptions. The appropriate dialog box automatically opens when you double-click an inserted step or exception for editing.

See Also ▶"Sequence Controller Instruction Set" on page 148

Editing Boolean Instruction Properties

When you double-click a Boolean step for editing, the appropriate properties dialog box opens. Each Boolean instruction's properties dialog box allows you to define output control properties, specify whether the instruction is interruptible by an exception, edit the Boolean expression, and enter descriptive comments. Boolean expressions such as conditional branches, also allow you to specify "timeout" and "goto" properties.

Edit "Wait for Bool" Properties This dialog box allows you to define a "wait for" Bool statement and specify output signal values.

Edit "Wait for Bool With Timeout" Properties This dialog box allows you to set up a "wait for" Boolean statement with an auto timeout. If the Boolean condition is met before the selected timer expires, the sequencer will continue to the next step. If the condition is not met and the timer expires, the sequencer will go to the specified step. The timer is started when the sequence enters this instruction step. The "with output" option specifies output signals that occur after the timer expires and transitions to the "Then Goto" step.

Conditional Branch (IfThenElse) This dialog box enables you to set up a conditional statement that checks a Boolean condition and then branches to an instruction step.

- The sequence will jump to the selected "Then Goto" step if the condition is true. The "With Output" option specifies output signal values that occur after the "Then Goto" transition.
- The sequence will jump to the selected "Else Goto" step if the expression is false. The "With Output" option specifies output signal values that occur after the "Else Goto" transition.

Edit "IfThenElse with Timeout" Properties This dialog box allows you to set up a conditional statement that checks a Boolean condition and the status of a timer.

- If the timer has expired, the sequencer will go to the "On Timeout Goto Sequencer step." The "with Output" option specifies output signal values that occur after the step transition.
- If the Boolean condition is met before the selected timer expires, the sequencer will go to the selected "Goto Sequencer step if no timeout and the Boolean expression is satisfied" step. The "With Output" option specifies output signal values that occur after the step transition.
- If neither the Boolean condition is met nor the selected timer has expired, the sequencer will go to the "Else Goto Sequencer step."

Note

This instruction checks the selected timer but does not start it. The timer must be started using a "Start Timer" instruction or supervisory logic.

See Also ▶"Sequence Controller Instruction Set" on page 148

Editing Boolean Expressions

The Boolean Expression Editor allows you to set up a Boolean expression using logic operators and a list of available inputs. You can enter the expression manually in the Expression text box at the top or select from the Logic Signal Pool and operators.

To edit a Boolean expression:

1. On the Sequence page, double-click the Boolean instruction that you want to edit.
2. In the Edit properties dialog box, click **Edit Boolean Expression**.

   You can enter the Boolean expression manually, if desired, by typing it into the Expression text box at the top and clicking OK. Otherwise, proceed with Steps 3-5.
3. Double-click a signal or bus from the Logic Signal Pool to add it to the expression.

   You can use the Filter input box below the signal pool to reduce the number of signals available. The signal pool only displays signals with the character string contained in the filter box.
4. Click the desired operator to add it to the expression.

   The following special vector operator is available for comparisons between buses and single-bit signals:

   - BITEXTEND (#) – Extends single bit to <wide> type vector. The BITEXTEND operator is only valid for single-bit signals.
5. Click **OK**.

The Boolean Expression Editor checks the expression for correctness and will issue a message if an error is encountered.

See Also ▶"Sequence Controller Instruction Set" on page 148

Editing Timeout Properties

The properties dialog boxes for "Wait for Timeout," "Start Timer," and "Stop Timer" enable you to select a defined timer, specify outputs, and specify whether the step can be interrupted by an exception.

All properties dialog boxes for sequence instructions that include timeouts—for example, the "Wait for Bool with Timeout"—allow you to select from a "Timeout" list of timers that have been defined. The properties dialog boxes for these instructions do not allow you to change the clock source and period for a selected timer. To do this, you must use the Timers sheet.

See Also ▶"Defining Timers" on page 151.

Editing Goto, NOP, and Halt Properties

The Edit "Goto" instruction properties dialog box allows you to specify the destination step for a GOTO (or branch) type of instruction, set the output signal values, enter a comment, and specify whether the instruction is interruptible by an exception.

The Edit NOP Properties dialog box allows you to enter a comment and specify that the NOP instruction is interruptible by an exception. A NOP step is essentially a single-cycle delay; it does not affect any of the outputs or the sequence of execution.

The Edit Halt Properties dialog box allows you to specify output values, enter a comment, and specify that the instruction is interruptible by an exception. A Halt step stops the execution of the sequence.

See Also ▶"Sequence Controller Instruction Set" on page 148

Editing Outputs

The Edit Output Properties dialog box enables you to select the outputs for a sequence instruction. It is immediately available when you double-click an Outputs instruction. For other instructions, it is accessible through the "Output Control" button or "With Output" button of the properties dialog box for the selected sequence.

Any single signal can be set to "Do not Modify," which preserves the signal value; or it can be set to a binary, hexadecimal, or decimal literal value.

Port groups can also be set to Increment, Decrement, Increment with Saturation, or Decrement with Saturation. These operations support counter implementations in the logic. The "with Saturation" instructions will saturate at

all 0s (decrement) or all 1s (increment). The decrement and increment operations will roll over.

See Also ▶"Creating a Port Group" on page 144

Editing Exceptions

You can add an exception for use in a control sequence by double-clicking the <end-of-exception-table> line in the Exceptions section of the Sequence page or selecting it and pressing the Insert key. Afterwards, double-click the inserted exception placeholder to set the properties.

In the Exception Properties dialog box, you can type the expression that will trigger the exception in the Expression text box at the top. Alternatively, click the **Edit** button to open the Boolean Expression Editor and select the signals and operators for the expression.

Select the step that the sequence will go to when the exception is encountered.

Then, in the Logic Signal Pool, you can select output signals or groups that will be modified when the exception occurs.

Note

Any output signals that are controlled by the exception expression are active at all times. These outputs will be modified, even if the sequence is currently in a non-interruptible step.

See Also ▶"Designing Control Sequences" on page 150

- "Editing Boolean Expressions" on page 153
- "Sequence Controller Instruction Set" on page 148

Entering Supervisory Equations

The supervisory equations define combinatorial and registered logic independent of the sequencer control logic. The supervisory equations are always active and execute in parallel to the sequencer logic.

To enter supervisory equations:

1. In the LogiBuilder Logic editor, select the **Supervisory** tab.
2. Insert an equation placeholder by double-clicking the **<end-of-supervisory-logic-table>** marker. Alternatively, highlight the marker and press the Insert key on your keyboard.
3. Double-click the equation placeholder to open the Supervisory Logic Equation Entry dialog box.

4. Select the output signal of the Supervisory equation. The output signal is chosen from the available pool, which can be reduced using the filter input. Output signal groups can also be assigned.
5. Choose the assignment type. A signal can be assigned combinatorially, as D flip-flop, or as asynchronous set or reset. Only signals that are already assigned in the sequence, or assigned as D flip-flop in other equations, can be assigned as asynchronous set or reset.
6. Edit the expression either directly in the expression or click Edit to use the Boolean Expression editor.
7. Click **OK** to complete the equation definition.

Note

The sequencing, exceptions, and supervisory equations are combined together during the compilation process. Outputs that are already assigned in the sequence cannot be assigned as D type in the logic equations. They can only be accessed by asynchronous reset or preset equations.

See Also ▶"Editing Boolean Expressions" on page 153

Copying and Pasting Sequences and Equations

The LogiBuilder Logic editor allows you to copy, cut, and paste control sequences, exceptions, and supervisory equations. If you are copying multiple instructions, they must be contiguous.

To copy or cut and paste in the Logic editor:

1. Select the sequences, exceptions, or equations that you want to copy or cut.

   Note

   The Begin Startup Sequence, Halt (end of program), <end of exceptions table>, and <end of supervisory table> cannot be included in your selection. If you include them, you will receive an error message.

2. Choose **Edit > Copy** or **Edit > Cut** or use the Ctrl+C or Ctrl+X keyboard shortcut.
3. Select the step, exception ID, or equation, where you wish to insert the copied items.
4. Choose **Edit > Paste** or use the Ctrl+V keyboard shortcut.

   The copied items are pasted before the selected step, exception, or equation. Branch targets will not be pasted along with the instruction or equation. These need to be re-entered manually.

   If you need to undo the action, chose **Edit > Undo** or use the Ctrl+Z keyboard shortcut.

See Also ▶"Sequence Controller Instruction Set" on page 148

Importing HDL Modules

The LogiBuilder Logic editor enables you to add an HDL module to a Platform Designer project. You must first add the source file to your Diamond project. Only one module can be imported into a Platform Designer project. To include multiple modules, you must structure a top-level module that you will pass to Platform Designer.

To import an HDL Module:

1. In Diamond's main window, choose **File > Add > Existing File**.
2. Navigate to the directory that contains the Verilog or VHDL source file, select the file and click **Add**.

   Diamond adds the file to the project and displays it in the Input Files folder of the File List view.
3. In Platform Designer, open the Logic editor and select the **Imported HDL** tab.
4. Select the **Enable Imported HDL** option. This will parse the input files for modules available to import.
5. Select the module from the Module Name menu.

   Platform Designer populates the File Location of the chosen module automatically. It also populates the Instance Name text box with the name of the module and adds the "Inst" extension. The left portion of the screen will be populated with any parameters available for definition in the imported HDL, as well as port information from the module.
6. If desired, rename the module instance by typing a new name in the Instance Name text box.
7. Assign values to the available HDL parameters.
8. For each port of the imported module, drag the desired signal from the Logic Signal Pool over to the Signal Name cell. The Logic editor's DRC checking will inform you if the signal you are trying to map is not compatible.

Building the Design

The Build view enables you to do a final design rule check, examine resource utilization, compile the design, assign pins, implement the design, and generate the merged JEDEC file. This view enables you to generate a test bench, including stimulus, for simulating your design. It also enables you to export a detailed configuration report.

Checking Design Rules

Design rules are checked automatically each time you issue a save command or run synthesis. Automatic design rule checking also prohibits you from entering or pasting invalid data into cells that have pre-defined ranges or numeric data types.

You can also use the Build view to run manual design rule check at any time.

To run manual design rule check:

- Click the **DRC** button at the top of the Build view.

  The results are displayed in the Output view of Diamond's main window.

Examining Resource Utilization

The Build view of Platform Designer displays a summary of the hardware resources that are being used by the current design configuration. It provides estimates of the consumption of device resource logic, major blocks, and I/Os of the current design. The Look Up Tables, Embedded Block RAM, and Programmable IOs area of the resource summary are updated after you run the Generate JEDEC step.

The Component Summary table at the top of the Build view indicates the status of the design components and programming options. The lower portion of the view provides a list of resources used in the design and the percentage of each resource consumed. At the bottom left corner, a green check mark or red X indicates the status of the implementation. When any options have been changed that affect the build output, the status will be marked "Not Current." You will then need to click the Compile button, to update the output, and complete the Pin Assignment and Generate JEDEC steps. If the implementation is successful, with no errors, the status will change to Current.

Compiling the Design

When you compile the design, Platform Designer generates the HDL code and synthesizes the logic design.

To compile the design:

- In the Build view of Platform Designer, click the **Compile** button.

  If you have unsaved changes in your design, the Save Modified Files dialog box will open. Select the .ptm file in the Modified File list and any others that you want to save and click **OK**.

  Platform Designer compiles the design and takes it through the synthesis and translation processes.

See Also "Examining Resource Utilization" on page 158

- "Exporting a Detailed Configuration Report" on page 159

Assigning Pins

After compiling the design in Platform Designer, you can open Diamond's Spreadsheet View to assign pins.

To assign pins:

1. In the Build view, click **Pin Assignments**.

   Spreadsheet View opens to the Port Assignments sheet.
2. Select the pin cells in the Pin column for the signals you want to assign.
3. Right-click the selected cells and choose **Assign Pins**.
4. Make your selections in the Assign Pins dialog box, and click **Assign Pins**.

See Also ▶"Assigning Signals in Spreadsheet View" on page 466 in Diamond online help.

Generating the JEDEC

The Build view enables you to generate a merged JEDEC file after pin assignment. This process merges the JEDEC of the FPGA with the ASC JEDECs.

To generate the merged JEDEC file:

- In the Build view, click **Generate Jedec**.

  When the process is completed, the Summary Status at the bottom of the Build view will be set to Current.

Exporting a Detailed Configuration Report

After the design has been successfully compiled and implemented, the Build view shows a green check mark, indicating that the status is current. This allows you to export a detailed report of the design's configuration.

To export a detailed configuration report:

- Select the Build view and click the **Export Configuration Report** button.

  Platform Designer automatically exports the report, named <project_name>_report.txt, to your project directory. You can view the report by following the link displayed after the report is exported.

  The report lists the components utilized in the design, along with the component settings for each feature. The report also includes full configuration information for the analog portion of the design.

Simulating the Design

After the design has been compiled successfully, you can perform functional simulation using Platform Designer and the Diamond Simulation Wizard. Platform Designer enables you to create a test bench file, based on the ASC configurations and specified connections, and it generates a stimulus file based on clock and non-clock data signals that you specify.

Generating a Stimulus and Test Bench File

After the design has been compiled successfully, you can specify the connections and signals for functional simulation. Platform Designer uses this information, plus the port-related information from the FPGA top-level HDL and the ASC configuration files, to generate a stimulus file and a test bench file.

To generate a stimulus file and a test bench file:

1. In the Build view of Platform Designer, click the **Generate Stimulus** button.

   The dialog box opens to the “Connections between FPGA and ASCs” page. The “From” and “To” sections, at the top, list all the FPGA ports and ASC signals that you can use to establish new board-level connections. The “Connection” section, at the bottom, displays the default hardware connections between the FPGA and all ASC devices in the design. These cannot be edited or removed.

2. Click the “**add connection**” button  on the bottom left.

   A new row is added at the bottom of the Connection section.

3. Scroll down to the newly added row, and then do the following:

   a. Select the desired signal in the “From” column in the top section and drag it to the “From Signal” cell in the newly created row at the bottom.

   b. Likewise, select the desired signal in the “To” column in the top section and drag it to the “To Signal” cell in the newly created row at the bottom.

   c. Type a name for the new net in the Net Name cell of the newly created row.

   Only a connection’s Net Name can be edited after you have selected the signals. To change the signals of a connection, you must remove the connection and replace it. To remove a connection, click any cell in the connection, and then click the X button.

4. Repeat Step 3 for each new connection you want to add.

5. Select the **Stimulus** tab, and then do the following:

   a. Select the **Clock** tab on the left to edit the stimulus of clock signals.

- Click the "**add clock stimulus**" button on the bottom left to add a clock. The system clock signal, ASC0_CLK, will be shown when this tab is opened, and it cannot be edited.
- Select the desired clock signal from the Signal Name drop-down menu, and then select an initial value, 1 or 0, from the Initial Value menu.
- In the Frequency or Period cell, type a value, select the measurement unit, and press **Enter**. The period is calculated automatically when you enter the frequency and vice versa.

b. Select the **Data** tab on the left to add stimulus for non-clock signals.

- Click the "**add data stimulus**" button on the bottom left to add a new row.
- Click inside the Signal Name column and choose a signal from the drop-down menu.
- The column following the Signal Name specifies the initial value. Click inside the cell and select the numerical type—Bin, Hex, or Dec—and then enter a value. Enter a value in each subsequent cell where the signal changes.

  You do not need to enter a value for each signal in each cell. Enter a value in a cell only when the value changes. The input value will be held at the last entered value if no entry is made in a cell.

  You can also select a bus and enter the values. The values can be entered as Binary, Decimal, or Hexadecimal for bus signals.
- Repeat the above steps to add stimulus for more non-clock signals.
- Add time entries for the additional columns in the top row of the display. You can select the time base to use as you enter each time value. The time entries are absolute time references rather than relative to each other. Therefore, the time values must be increasing from left to right.

  When you add a time entry in the last column on the right, a new column gets added on the right, allowing you to enter an additional time value if desired.

  To add or remove a column, right-click the time entry top row where you want to insert or remove a column and choose Insert Before, Insert After, or Remove. Make sure that the time entry cell is not open for editing before you right-click. If it is open for editing, press the Enter key, and then right-click to access the pop-up menu.

6. Click **OK**.

Simulating the Design

When you generate the stimulus for the design, the following four files are automatically created:

<project_name>_design.v This file serves as a wrapper of the MachXO2HC/HE, MachXO3D, MachXO3LF, ECP5U/UM and ASC for functional simulation

<project-name>_test.v This file serves as the top-level file for functional simulation

<project-name>_design_post.v This file serves as a wrapper of the MachXO2HC/HE, MachXO3D, MachXO3LF, and ASC for timing simulation.

<project-name>_test_post.v This file serves as the top-level file for timing simulation

You can use Diamond's Simulation Wizard  to create the simulation project for the Platform Manager 2 or MachXO2HC/HE, MachXO3D, MachXO3LF, ECP5U/UM design and export the simulation project into ModelSim Lattice FPGA Edition or Aldec Active-HDL full version. After opening the Simulation Wizard, follow the on-screen prompts to create the simulation script.

When you create the simulation project in the Diamond Simulation Wizard and click **Finish**, the ModelSim software will start automatically. If "Run Simulation" (the default value) was selected in the Simulation Wizard, the project will be loaded into the simulator. The project files will be compiled, and the simulation waveform window will open with all the top-level signal names. You can add signal names from within the design to the waveform window if you wish to see other signals from your design. See the ModelSim documentation for more information about adding signal names to the waveform window.

The simulation will run for a period of 1 µs after it is loaded and compiled. You can run it for additional time by using the Run command in the ModelSim console. To run the simulation for 100 µs, type the command `<run 100 us>` and press the Enter key. The length of time to run your simulation will depend upon the design and the stimulus file you created.

For each ASC configuration, the Simulation Wizard will copy the following files to the simulation folder:

- cfg_eeprom.hex
- flut_eeprom.hex
- i2caddress.hex
- trim_eeprom.hex

The Simulation Wizard will also copy the memory file (.mem) for the "VID table" and the MICO "scratchpad" and "prom" files.

The Simulation Wizard will not copy any memory or data file from an imported HDL module. If you have imported an HDL module into your Platform Designer project, you will need to copy it manually into the simulation folder.

See Also ▶"Simulation in Diamond" on page 310 in Diamond online help.

Working with Power Calculator

When using Platform Designer with Power Calculator, the most accurate power calculation is accomplished when using Power Calculator in integrated mode within your projects.

When calculating power for Platform Manager 2 and MachXO2HC/HE, MachXO3D, MachXO3LF, ECP5U/UM designs using an ASC device, the Diamond integrated Power Calculator tool imports a data file generated by the Platform Designer software. This data file is generated in the current implementation directory and should not be edited by the user.

Platform Designer calculates the power mode for each ASC used and generates a data file containing each ASC, the power used by that ASC, and the power supply used (if relevant). Refer to the description of the "ASC" on page 767 page in Diamond Power Calculator online help. The data file contains both external ASC devices used and the ASC internal component in the Platform Manager 2 device.

The power consumed by an ASC device is determined by the number of HVOUTs used in the design. The data file, which is generated from Platform Designer and which has an .apw suffix, includes the number of HVOUTs used and the power model equation. The data file lists all ASCs and the power used by each and is located in the design implementation directory. This file is read by Power Calculator and then the additional power from the ASC devices is added to the total power displayed. This data is displayed in the Power Calculator Power Summary tab, Power by Block (W) section and the ASC page.

Power calculation for ASCs is different than for other Lattice devices. To calculate power consumption for ASCs, the following equations are used:

$$Current = I_O + I_{HVOUT} \times \#HVOUT$$

$$Power = V_{CC} \times Current$$

$$Power = V_{CC} \times (I_O + I_{HVOUT} \times \#HVOUT)$$

There are four variables:

$V_{CC} = 3.6V$

$I_O = 23mA$

$I_{HVOUT} = 2mA$

$\#HVOUT =$ Number of HV Outputs used

The above calculations are only used when Power Calculator is used in the integrated mode or used with a PCF file generated in this mode. To understand the different methods power is calculated refer to the list below.

Power Calculator Integrated mode

- Calculates power from multiple ASC devices for Platform Manager 2, or MachXO2HC/HE with external ASCs, MachXO3D with external ASCs, MachXO3LF with external ASCs, or ECP5U/UM with external ASCs.

Power Calculator standalone mode

- Calculates power only from the internal ASC in the Platform Manager 2 device. ASC power is estimated as a fixed value, not a variable amount based on the number of HVOUTs used.
- Can correctly calculate ASC power based on HVOUTs used only if opened with a PCF file generated by Power Calculator in integrated mode instead of loading NGD database.

Power Estimator standalone mode

- Calculates power only from the internal ASC in the Platform Manager 2 device. ASC power is estimated as a fixed value, not a variable amount based on the number of HVOUTs used.
- Can correctly calculate ASC power based on HVOUTs used only if opened with a PCF file generated by Power Calculator in integrated mode.

See Also ▶"Analyzing Power Consumption" on page 756 in Diamond online help.

Chapter 4

Entering the Design

You can enter your design in several different ways and you can use the ways in almost any combination. Lattice Diamond accepts, and helps create, Verilog, VHDL, schematic, and EDIF files. Design files can be created within Diamond or outside using third-party tools. Diamond also helps you use a variety of pre-made modules, ranging from simple functions to whole processors.

Here, “entering the design” refers to describing the logic of the design. Setting specifications to control synthesis or simulation or to control how the design is implemented in hardware are treated as a separate topic. (See “Applying Design Constraints” on page 360.) You will almost always want to at least start entering your design before setting constraints.

Verilog and VHDL The hardware design languages (HDL) Verilog and VHDL are the usual ways to describe complex logic designs. You can create HDL files using your preferred HDL editor and then add them to your Diamond design project, or you can create or edit them within Diamond. See “HDL Design Entry” on page 167.

Modules Modules are functional bits of design that can be re-used wherever that function is needed. To help your design along, Lattice Semiconductor provides a variety of modules for common functions. They are optimized for Lattice device architectures and can be customized. Use these modules to speed your design work and to get the most effective results. See “Designing with Modules” on page 190.

Lattice Propel Lattice Propel helps you produce system-on-chip (SoC) designs using a RISC-V processor. Propel provides the CPU, memory, and peripheral buses as customizable modules with an easy drag-and-drop interface. Propel also provides an industry-standard integrated development environment (IDE) for writing and debugging software. Propel is a standalone tool that you need to download and install separately. See “Designing with Lattice Propel” on page 239.

LatticeMico Platforms A special kind of module is the LatticeMico platform. LatticeMico platforms are full-featured 32-bit processors and 8-bit controllers plus peripheral components created with LatticeMico System, an optional add-on to Diamond. LatticeMico System provides the processor, the controller, and a variety of peripheral components. With it, you can customize the platform and debug C and C++ code. See “Designing with LatticeMico Platforms” on page 241.

Schematic Design An alternative to HDL is schematic design, in which you graphically lay out modules and other components and wire them together. Schematic is a very visual method but lacks the abstraction possible in an HDL. Schematic design may be better suited for smaller designs or smaller portions of a design. It's also a good way to lay out the top level of a design. The detailed workings of lower levels could then be done in an HDL. See "Schematic Design Entry" on page 243.

EDIF Actually, EDIF (Electronic Design Interchange Format) is not a way to enter a design but is instead a way to exchange design data between different EDA systems. But EDIF files can be added to a Diamond design project like input files such as Verilog and VHDL. You may want to add EDIF files because you prefer to use another synthesizer or because you have EDIF files of modules that you want to re-use from another project. Diamond supports EDIF 2 0 0. See "Importing an Existing Source File into a Project" on page 36.

HDL Design Entry

Diamond supports HDL design including pure VHDL design, pure Verilog design, and mixed VHDL and Verilog HDL design. You can use the integrated Source Editor to create and edit your HDL source files and any text-based files.

Source Editor is a text entry tool of Lattice Diamond. You can use this tool to create and edit text-based files, such as HDL files, preference files, script files, test files, and project documentation files. Like many other advanced editing tools, it supports multiple file editing, wheel mouse scrolling, as well as popup menus for cut, copy, and paste operations. If a data source changes since the view was initialized, the view will detect this condition and provide you an option to refresh it.

Designed particularly for HDL file editing, Source Editor highlights the std_logic keywords and displays different HDL syntax elements with different colors. The color scheme can be customized in the Options dialog (**Tools > Options** from the main window) of the Diamond main window, the **Source Editor > Colors** section. Also, Source Editor enables you to check correct pairing of parenthesis constructs and offers rich template features.

Note

To avoid errors, observe the following file naming rules:

- Do not leave blank spaces in file names when you save source files in Source Editor, otherwise the files could fail to generate the database.
- Begin file names with an alpha character (letter) rather than a numeric character (number).
- Use underscores (_) rather than leave blanks in file names.
- Unicode characters are not supported.
- Keep file names, and paths to files, short as possible.
- Linux operating system is case-sensitive. Use lower-case letters in file names to avoid issues with case-sensitive operating systems.

If you are going to use Lattice Synthesis Engine (LSE) to synthesize the design, there are helpful Verilog and VHDL coding tips in “Coding Tips for Lattice Synthesis Engine (LSE)” on page 173.

See Also ▶“Running Source Editor” on page 167

- “Viewing Logs and Reports” on page 101

Running Source Editor

You can run Source Editor in several ways.

To run Source Editor, do one of the following:

- From the File List view, double-click a language file—for example, filename.vhd
- From the Diamond main window, choose **File > New > File**. In the New File dialog, choose a text-based source, for example Verilog Files. Fill in the File name and Location, click **New**.

Creating HDL Source Files in Source Editor

You can create an HDL source file in Source Editor.

To create an HDL source in Source Editor:

1. From the Diamond main window, choose **File > New > File**. In the New File dialog, choose Verilog Files or VHDL Files from the Source Files list.
2. In the pop up New File dialog, fill in the File name and Location, choose the file extension in the Ext field.
3. Check the **Add to project** option if you want to add this source to the current project.
4. click **New**.
5. In the pop up Source Editor, you can enter the text. When finished editing, click **File > Save** from the Diamond main window.

Tip

You can detach Source Editor from the Diamond main window by clicking the Detach Tool icon on the upper right corner of Source Editor. If you want to attach Source Editor back to the main window, click the Attach Window icon on the upper right corner of Source Editor window, or choose **Window > Attach Window** from Source Editor.

Editing Text Files

This section describes the editing functions in Source Editor.

Editing a Column of Text

You can cut, copy, and paste a column of selected text in Source Editor. You can also use the Tab key to increase the left indention of the selected column.

To cut, copy, and paste a column of text:

1. Hold down **Alt**, and then click and hold the left mouse button down to form a rectangle that you want to edit.

   IMPORTANT!

   This feature does not work on a Linux platform, because the **Alt** key is occupied by the Linux system.

2. Use the right-click commands or the commands on the Edit menu to cut, copy, and paste the selected text.

To indent a column of text:

1. Place the cursor in front of the text you want to indent.
2. Choose **Edit > Advanced > Indent**. Or, press **Tab**.

   Source Editor increases the indent of the selected text column by one tab size. By "Matching Braces, Parenthesis, and Brackets" on page 169, you can change the position of the indent.

To uppercase a selection:

1. Highlight the text you want to change to uppercase.
2. Choose **Edit > Advanced > Uppercase Selection**. You will see the highlighted text become uppercase.

To lowercase a selection:

1. Highlight the text you want to change to lowercase.
2. Choose **Edit > Advanced > Lowercase Selection**. You will see the highlighted text become lowercase.

To comment text:

1. Highlight the text you want to add comment, or place the cursor in front of the text.
2. Choose **Edit > Advanced > Comment/Uncomment**. You will see "--" put ahead of the text (VHDL file only). You can add comment after "--" (VHDL file only).

Matching Braces, Parenthesis, and Brackets

Source Editor monitors brace ({}), parenthesis (()), and bracket ([]) constructs. You can match an opening brace (or bracket or parenthesis) with a closing one, and vice versa.

To jump to the matching brace, parenthesis, or bracket:

1. In Source Editor, place the editing cursor adjacent to an opening or closing parenthesis (or bracket or brace).
2. Choose **Edit > Advanced > Match Brace**.

The cursor will move to the corresponding opening or closing parenthesis (or bracket or brace). This allows you the check for the balanced braces around functions and statements.

You can use the **Edit > Advanced > Select to Brace** command to force the matching brace, parenthesis, or brackets to be highlighted along with the text in between the two braces, parenthesis, or brackets,

Using Bookmarks

You can insert bookmarks into the source file to mark places where you need to enter variables. Afterwards, each time you load the file, you can use the **Edit > Bookmarks** command to jump to those marks and enter or edit variables.

To insert bookmarks in a file:

1. In Source Editor, open the file where you want to insert bookmarks.
2. Click the place you want to insert a mark.
3. Choose **Edit > Bookmarks > Toggle Bookmark**.

   A mark is inserted at the beginning of the line where the cursor points.
4. Repeat step 2 and 3 to add more marks if so desired.

To find marks in a file:

1. In Source Editor, open the file where you want to find marks.
2. Choose **Edit > Bookmarks > Previous Bookmark**, or **Edit > Bookmarks > Next Bookmark**.

   The cursor jumps to one mark or the other. You can start typing from the desired marked place to enter or edit the variable.
3. Repeat step 2 to find, enter, or edit more marks if so desired.

Using AutoComplete

When you type a keyword into Source Editor, choose **Edit > Advanced > AutoComplete**. A list of the matching values will be presented. You can then:

- Choose the first matching value by pressing **Enter** directly.
- Choose an alternate matching value by using the up/down arrow keys and then press **Enter**.
- Continue typing the rest of the value. If no matching value is found, nothing will be entered.

Using Templates

Source Editor provides templates for creating VHDL, Verilog, and constraint files. Templates increase the speed and accuracy of design entry. You can also create your own templates. Remember to replace dummy variables in the template with your own logic.

You can use Source Editor's **View > Template Editor** menu to locate and access template files. After clicking this menu, you can see the Template Editor pane located in the upper left of Source Editor. Templates are available for:

- Verilog
- VHDL
- SDC (Synopsys Design Constraints. Can also be used for FDC files.)
- LDC (Constraints for LSE.)

Each category has folders for System Templates and User Templates. System Templates holds the standard templates that come with Diamond. For Verilog and VHDL, these include commands for instantiating PMI modules and common elements of the language. For SDC and LDC, System Templates holds templates for the supported constraint statements. User Templates allows you to create your own templates for future use.

See Also ▶"Lattice Module Reference Guide" on page 1400

- "Lattice Synthesis Engine (LSE) Constraints" on page 1339

Inserting a Template into Your Source File

You can insert templates into a source file you are editing with Source Editor.

To insert a template into your source file:

1. Open the source file in Source Editor. See "Running Source Editor" on page 167.
2. Choose **View > Template Editor**.

   Two additional panes open in Source Editor. The upper pane shows the list of template folders. The lower pane is blank.
3. Place the cursor where you want to insert a template. This should be in a blank line.
4. Select the template you want to use in the template list.

   The code of the template appears in the lower pane.
5. Right-click the template name and choose **Insert to text**.

   The selected template is inserted into the file after the cursor.
6. Be sure to replace all placeholders with real values. Placeholders are terms between angle brackets such as "<event_expression>."

7. If you inserted a PMI template, also add parameter values and I/O signal names. For information about allowed parameter values, see the section on the module in the “Lattice Module Reference Guide” on page 1400.
8. If needed, make further changes such as adding code to execute within a case statement.

See Also ▶“Instantiating a PMI Module” on page 201

Creating a New Template

You can create new templates for your own use and save them into the User Templates folders. The new templates will be available for all implementations in the design project.

You can also create sub-folders to organize the templates that you create. If you plan on creating many templates, think about the organization before you start. There is no easy way to move templates once they have been created.

To create a new template:

1. Open a source file in Source Editor. This can be a file that you will use the new template in or a blank file that you will delete after creating the template. See “Running Source Editor” on page 167.
2. Choose **View > Template Editor**.

   Two additional panes open in Source Editor. The upper pane shows the list of template folders. The lower pane is blank.
3. In Template Editor, expand the appropriate folder so that its User Templates folder is showing. Every top level folder has a User Templates folder. If the User Templates folder already has contents, expand it so you can see the contents.
4. If you want to create a sub-folder, right-click **User Templates** or one of its sub-folders and choose **New Folder**. Then right-click the new folder and choose **Rename**. Type the desired folder name and press **Enter**.
5. Right-click **User Templates** or one of its sub-folders and choose **New Template**.

   A new template icon appears under the folder.
6. Right-click the new template icon and choose **Rename**. Type in the desired template name and press **Enter**.
7. Type the code for your template in the lower pane of Template Editor.

   The new template is automatically saved.

Deleting a User Template

You can only delete templates from a **User Templates** folder.

There is no undo command. Be sure you have the right template and want to permanently remove it.

To delete a template:

1. Open a source file in Source Editor. This can be a file that you are using or a blank file that you will delete after creating the template. See “Running Source Editor” on page 167.
2. Choose **View > Template Editor**.

   Two additional panes open in Source Editor. The upper pane shows the list of template folders. The lower pane is blank.
3. In Template Editor, expand the folders so that the desired folder or template is showing.

   Caution

   Deleting a folder also deletes all templates under the folder.
4. Right-click that template or folder and choose **Delete**.

Coding Tips for Lattice Synthesis Engine (LSE)

If you are going to use LSE to synthesize the design, the following coding tips may help. Mostly the tips are about writing code so that blocks of memory are “inferred”: that is, automatically implemented using programmable function units (PFU) or embedded block RAM (EBR) instead of registers. There are also tips about inferring types of I/O ports and avoiding conflicts between VHDL packages.

About VHDL Coding

LSE tends to apply the VHDL specification strictly, sometimes more strictly than other synthesis tools. Following are some coding practices that can cause problems with LSE:

ieee.std_logic_signed or unsigned When preparing VHDL code for LSE, you can include either:

USE ieee.std_logic_signed.ALL;

or:

USE ieee.std_logic_unsigned.ALL;

DO NOT include both statements. Code with both signed and unsigned packages could fail to synthesize because operators would have multiple definitions. Some synthesis tools may allow this but LSE does not.

Strict Variable Typing LSE is stricter about variable type requirements than some other synthesis tools. A std_logic_vector signal cannot be assigned to a std_logic signal and an unsigned type cannot be assigned to a std_logic_vector signal. For example:

```
din : in  unsigned (data_width - 1 downto 0);
dout : out std_logic_vector (data_width - 1 downto 0));
...
dout <= din; -- Illegal, mismatched assignment.
```

Such mismatched assignments generate errors that stop synthesis.

About Inferring Memory

There are two ways to produce RAM and ROM in a design. You can write code for the design so that the synthesis tool infers the memory or you can instantiate predefined IPexpress or PMI memory modules.

Inferring memory means that LSE, based on aspects of the code, implements a block of memory using programmable function units (PFU) or embedded block RAM (EBR)—PFU for small memories, EBR for large—instead of registers. LSE can infer synchronous RAM that is:

- single-port, pseudo dual-port, or true dual-port
- with or without asynchronous reset of the output
- with or without write enables
- with or without clock enables

LSE can also infer synchronous ROM.

One of the advantages of inferring memory is that the design is portable to almost any FPGA architecture. PMI modules, on the other hand, are optimized for Lattice FPGAs and IPexpress modules are optimized for just one FPGA family. Also, because the inferred memory is not a black box, there's full access for tools and to timing and area data.

However, there may be less efficient use of the FPGA architecture. Also, not all kinds of memory can be inferred. In these situations you may prefer to use IPexpress or PMI modules. With these modules you can also use Memory Generator to easily create memory initialization files. See "Designing with Modules" on page 190.

The following sections describe how to write code to infer different kinds of memory with LSE.

See Also

- "Inferring RAM" on page 174
- "Inferring ROM" on page 184

Inferring RAM

The basic inferred RAM is synchronous. It can have synchronous or asynchronous reads and can be either single- or dual-port. You can also set initial values. Other features, such as resets and clock enables, can be added

as desired. The following text lists the rules for coding inferred RAM. Following that, Figure 5 on page 175 (Verilog) and Figure 6 on page 176 (VHDL) show the code for a simple, single-port RAM with asynchronous read.

To code RAM to be inferred, do the following:

- Define the RAM as an indexed array of registers.
- To control how the RAM is implemented (with distributed or block RAM), consider adding the syn_ramstyle attribute. See "syn_ramstyle" on page 1381.
- Control the RAM with a clock edge and a write enable signal.
- For synchronous reads, see "Inferring RAM with Synchronous Read" on page 177.
- For single-port RAM, use the same address bus for reading and writing.
- For dual-port RAM, pseudo and true, see "Inferring Dual-Port RAM" on page 179.
- If desired, assign initial values to the RAM as described in "Initializing Inferred RAM" on page 183.

Figure 5: Simple, Single-Port RAM in Verilog

```
module ram (din, addr, write_en, clk, dout);
  parameter addr_width = 8;
  parameter data_width = 8;
  input [addr_width-1:0] addr;
  input [data_width-1:0] din;
  input write_en, clk;
  reg [data_width-1:0] mem [(1<<addr_width)-1:0];
    // Define RAM as an indexed memory array.

  always @(posedge clk) // Control with a clock edge.
  begin
    if (write_en) // And control with a write enable.
      mem[(addr)] <= din;
  end
  assign dout = mem[addr];
endmodule
```

See Also

- "About Verilog Blocking Assignments" on page 185

Figure 6: Simple, Single-Port RAM in VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
generic (
  addr_width : natural := 8;
  data_width : natural := 8);
port (
  addr : in  std_logic_vector (addr_width - 1 downto 0);
  write_en : in  std_logic;
  clk : in  std_logic;
  din : in  std_logic_vector (data_width - 1 downto 0);
  dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
  type mem_type is array ((2** addr_width) - 1 downto 0) of
    std_logic_vector(data_width - 1 downto 0);
  signal mem : mem_type;
    -- Define RAM as an indexed memory array.
begin
  process (clk)
  begin
    if (clk'event and clk = '1') then --Control with clock edge
      if (write_en = '1') then -- Control with a write enable.
        mem(conv_integer(addr)) <= din;
      end if;
    end if;
  end process;
  dout <= mem(conv_integer(addr));
end rtl;
```

Inferring RAM with Synchronous Read

For synchronous reads, add a register for the read address or for the data output. Load the register inside the procedure or process that is controlled by the clock. See the following examples. They show the simple RAM of Figure 5 on page 175 (for Verilog) and Figure 6 on page 176 (for VHDL) modified for synchronous reads. Changes are in bold text.

Verilog Examples

Figure 7: RAM with Registered Output in Verilog

```
module ram (din, addr, write_en, clk, dout);
  parameter addr_width = 8;
  parameter data_width = 8;
  input [addr_width-1:0] addr;
  input [data_width-1:0] din;
  input write_en, clk;
  output [data_width-1:0] dout;
  reg [data_width-1:0] dout; // Register for output.
  reg [data_width-1:0] mem [(1<<addr_width)-1:0];

  always @(posedge clk)
  begin
    if (write_en)
      mem[(addr)] <= din;
    dout = mem[addr]; // Output register controlled by clock.
  end
endmodule
```

Figure 8: RAM with Registered Read Address in Verilog

```
module ram (din, addr, write_en, clk, dout);
  parameter addr_width = 8;
  parameter data_width = 8;
  input [addr_width-1:0] addr;
  input [data_width-1:0] din;
  input write_en, clk;
  output [data_width-1:0] dout;
  reg [data_width-1:0] raddr; // Register for read address.
  reg [data_width-1:0] mem [(1<<addr_width)-1:0];

  always @(posedge clk)
  begin
    if (write_en)
    begin
      mem[(addr)] <= din;
    end
    raddr <= addr; // Read addr. register controlled by clock.
  end
  assign dout = mem[raddr];
endmodule
```

VHDL Examples

Figure 9: RAM with Registered Output in VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
generic (
  addr_width : natural := 8;
  data_width : natural := 8);
port (
  addr : in  std_logic_vector (addr_width - 1 downto 0);
  write_en : in  std_logic;
  clk : in  std_logic;
  din : in  std_logic_vector (data_width - 1 downto 0);
  dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
  type mem_type is array ((2** addr_width) - 1 downto 0) of
    std_logic_vector(data_width - 1 downto 0);
  signal mem : mem_type;
begin
  process (clk)
  begin
    if (clk'event and clk = '1') then
      if (write_en = '1') then
        mem(conv_integer(addr)) <= din;
      end if;
    end if;
    dout <= mem(conv_integer(addr));
      -- Output register controlled by clock.
  end process;
end rtl;
```

Figure 10: RAM with Registered Read Address in VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
generic (
  addr_width : natural := 8;
  data_width : natural := 8);
port (
  addr : in  std_logic_vector (addr_width - 1 downto 0);
  write_en : in  std_logic;
  clk : in  std_logic;
  din : in  std_logic_vector (data_width - 1 downto 0);
  dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
  type mem_type is array ((2** addr_width) - 1 downto 0) of
    std_logic_vector(data_width - 1 downto 0);
  signal mem : mem_type;
begin
  process (clk)
  begin
    if (clk'event and clk = '1') then
      if (write_en = '1') then
        mem(conv_integer(addr)) <= din;
      end if;
      raddr <= addr;
        -- Read address register controlled by clock.
    end if;
  end process;
  dout <= mem(conv_integer(raddr));
end rtl;
```

See Also

- “Inferring RAM” on page 174

Inferring Dual-Port RAM

For dual-port RAM, pseudo or true:

- Use two address buses.
- If the design does not simultaneously read and write the same address, add the syn_ramstyle attribute with the no_rw_check value to minimize overhead logic.
- If writing in Verilog, use non-blocking assignments as described in “About Verilog Blocking Assignments” on page 185.

The following examples are based on the simple RAM of Figure 5 on page 175 (for Verilog) and Figure 6 on page 176 (for VHDL).

Verilog Examples

Figure 11: Pseudo Dual-Port RAM in Verilog

```
module ram (din, write_en, waddr, wclk, raddr, rclk, dout);
  parameter addr_width = 8;
  parameter data_width = 8;
  input [addr_width-1:0] waddr, raddr;
  input [data_width-1:0] din;
  input write_en, wclk, rclk;
  output [data_width-1:0] dout;
  reg [data_width-1:0] dout;
  reg [data_width-1:0] mem [(1<<addr_width)-1:0]
    /* synthesis syn_ramstyle = "no_rw_check" */ ;

  always @(posedge wclk) // Write memory.
  begin
    if (write_en)
      mem[waddr] <= din; // Using write address bus.
  end
  always @(posedge rclk) // Read memory.
  begin
    dout <= mem[raddr]; // Using read address bus.
  end
endmodule
```

Figure 12: True Dual-Port RAM in Verilog

```
module ram (dina, write_ena, addra, clka, douta,
  dinb, write_enb, addrb, clkb, doutb);
  parameter addr_width = 8;
  parameter data_width = 8;
  input [addr_width-1:0] addra, addrb;
  input [data_width-1:0] dina, dinb;
  input write_ena, clka, write_enb, clkb;
  output [data_width-1:0] douta, doutb;
  reg [data_width-1:0] douta, doutb;
  reg [data_width-1:0] mem [(1<<addr_width)-1:0]
    /* synthesis syn_ramstyle = "no_rw_check" */ ;

  always @(posedge clka) // Using port a.
  begin
    if (write_ena)
      mem[addra] <= dina; // Using address bus a.
    douta <= mem[addra];
  end
  always @(posedge clkb) // Using port b.
  begin
    if (write_enb)
      mem[addrb] <= dinb; // Using address bus b.
    doutb <= mem[addrb];
  end
endmodule
```

VHDL Examples

Figure 13: Pseudo Dual-Port RAM in VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
generic (
  addr_width : natural := 8;
  data_width : natural := 8);
port (
  write_en : in  std_logic;
  waddr : in  std_logic_vector (addr_width - 1 downto 0);
  wclk : in  std_logic;
  raddr : in  std_logic_vector (addr_width - 1 downto 0);
  rclk : in  std_logic;
  din : in  std_logic_vector (data_width - 1 downto 0);
  dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
  type mem_type is array ((2** addr_width) - 1 downto 0) of
    std_logic_vector(data_width - 1 downto 0);
  signal mem : mem_type;
  attribute syn_ramstyle: string;
  attribute syn_ramstyle of mem: signal is "no_rw_check";
begin
  process (wclk) -- Write memory.
  begin
    if (wclk'event and wclk = '1') then
      if (write_en = '1') then
        mem(conv_integer(waddr)) <= din;
          -- Using write address bus.
      end if;
    end if;
  end process;
  process (rclk) -- Read memory.
  begin
    if (rclk'event and rclk = '1') then
      dout <= mem(conv_integer(raddr));
        -- Using read address bus.
    end if;
  end process;
end rtl;
```

Figure 14: True Dual-Port RAM in VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
generic (
  addr_width : natural := 8;
  data_width : natural := 8);
port (
  addra : in  std_logic_vector (addr_width - 1 downto 0);
  write_ena : in  std_logic;
  clka : in  std_logic;
  dina : in  std_logic_vector (data_width - 1 downto 0);
  douta : out std_logic_vector (data_width - 1 downto 0);
  addrb : in  std_logic_vector (addr_width - 1 downto 0);
  write_enb : in  std_logic;
  clkb : in  std_logic;
  dinb : in  std_logic_vector (data_width - 1 downto 0);
  doutb : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
  type mem_type is array ((2** addr_width) - 1 downto 0) of
    std_logic_vector(data_width - 1 downto 0);
  signal mem : mem_type;
  attribute syn_ramstyle: string;
  attribute syn_ramstyle of mem: signal is "no_rw_check";
begin
  process (clka) -- Using port a.
  begin
    if (clka'event and clka = '1') then
      if (write_ena = '1') then
        mem(conv_integer(addra)) <= dina;
          -- Using address bus a.
      end if;
      douta <= mem(conv_integer(addra));
    end if;
  end process;
  process (clkb) -- Using port b.
  begin
    if (clkb'event and clkb = '1') then
      if (write_enb = '1') then
        mem(conv_integer(addrb)) <= dinb;
          -- Using address bus b.
      end if;
      doutb <= mem(conv_integer(addrb));
    end if;
  end process;
end rtl;
```

See Also

- “Inferring RAM” on page 174

Initializing Inferred RAM

Create initial values for inferred RAM in the usual ways for initializing memory.

Verilog In Verilog, initialize RAM with the standard $readmemb or $readmemh tasks in an initial block. Create a separate file with the initial values in either binary or hexadecimal form. For example, to initialize a RAM block named “ram”:

```
reg [7:0] ram [0:255];
initial
begin
  $readmemh ("ram.ini", ram);
end
```

The data file has one word of data on each line. The data needs to be in the same order in which the array was defined. That is, for “ram [0:255]” the data starts with address 0; for “ram [255:0]” the data starts with address 255. The ram.ini file might start like this:

```
0A /* Address 0 */
23
5C
...
```

VHDL In VHDL, initialize RAM with either signal declarations or variable declarations. Define an entity with the same ports and architecture as the memory. Use this entity in either a signal or variable statement with the initial values as shown below.

For example, to initialize a RAM block named “ram,” define an entity such as:

```
entity ram_init is
port (
  clk : in std_logic;
  addr : in std_logic_vector(7 downto 0);
  din : in std_logic_vector(7 downto 0);
  we : in std_logic;
  dout : out std_logic_vector(7 downto 0));
end;
architecture arch of ram_init is
  type ram_init_arch is array(0 to 255)
  of std_logic_vector (7 downto 0);
```

Then use the entity in a signal statement:

```
signal ram : ram_init_arch := (
"00001010",
"00100011",
"01011100",
...
others => (others => '0'));
```

Or use the entity in a variable statement:

```
variable ram : ram_init_arch := (
```

```
1 => "00001010",
...
others => (1=>'1', others => '0'));
```

See Also

- “Inferring RAM” on page 174

Inferring ROM

To code ROM to be inferred, do the following:

- Define the ROM with a case statement or equivalent if statements.
- Assign constant values, all of the same width.
- Assign values for at least 16 addresses or half of the address space, whichever is greater. For example, if the address has 6 bits, the address space is 64 words, and at least 32 of them must be assigned values.
- To control how the ROM is implemented (with distributed or block ROM), consider adding the syn_romstyle attribute. See “syn_romstyle” on page 1385.

Figure 15: ROM Inferred with Case Statement in Verilog

```
module rom(data, addr);
  output [3:0] data;
  input [4:0] addr;
  always @(addr) begin
    case (addr)
       0 : data = 'h4;
       1 : data = 'h9;
       2 : data = 'h1;
       ...
      15 : data = 'h8;
      16 : data = 'h1;
      17 : data = 'h0;
      default : data = 'h0;
    endcase
  end
endmodule
```

Figure 16: ROM Inferred with If Statement in VHDL

```
entity rom is
port (addr : in std_logic_vector(4 downto 0);
  data : out std_logic_vector(3 downto 0) );
end rom;

architecture behave of rom is
begin
  process(addr)
  begin
    if    addr =  0 then data <= "0100";
    elsif addr =  1 then data <= "1001";
    elsif addr =  2 then data <= "0001";
    ...
    elsif addr = 15 then data <= "1000";
    elsif addr = 16 then data <= "0001";
    elsif addr = 17 then data <= "0000";
    else                 data <= "0000";
    end if;
  end process;
end behave;
```

About Verilog Blocking Assignments

LSE support for Verilog blocking assignments to inferred RAM and ROM, such as “ram[(addr)] = data;,” is limited to a single such assignment. Multiple blocking assignments, such as you might use for true dual-port RAM (see Figure 17 on page 185), or a mix of blocking and non-blocking assignments are not supported. Instead, use non-blocking assignments (<=). See Figure 18 on page 186.

Figure 17: Example of RAM with Multiple Blocking Assignments (Wrong)

```
always @(posedge clka)
begin
  if (write_ena)
    ram[addra] = dina; // Blocking assignment A
  douta = ram[addra];
end
always @(posedge clkb)
begin
  if (write_enb)
    ram[addrb] = dinb; // Blocking assignment B
  doutb = ram[addrb];
end
```

Figure 18: Example Rewritten with Non-blocking Assignments (Right)

```
always @(posedge clka)
begin
  if (write_ena)
    ram[addra] <= dina;
  douta <= ram[addra];
end
always @(posedge clkb)
begin
  if (write_enb)
    ram[addrb] <= dinb;
  doutb <= ram[addrb];
end
```

About Verilog Generate Blocks

If a module has multiple generate blocks, LSE requires that they each block have a different name. Assign names with begin statements. See Figure 19 on page 186.

Figure 19: Generate Blocks with Assigned Names

```
generate
begin: R1_gen // Name first generate block.
  genvar i1;
  for (i1=0; i1<=15; i1=i1+1)
    regset R1 (.clk(clk[i1]),.q(q[i1]),.d(d[i1]));
end
endgenerate

generate
begin: IR0_gen // Name second generate block.
  genvar i;
  for (i=0; i<=7; i=i+1)
    ioreg IR0 (.clk(clk[0]),.reset(rst[i]),.q(q[i]),.d(d[i]));
end
endgenerate
```

Inferring I/O

To specify types of I/O ports, follow these models.

Verilog

Open Drain:

```
output <port>;
wire <output_enable>;
assign <port> = <output_enable> ? 1'b0 : 1'bz;
```

Bidirectional:

```
inout <port>;
wire <output_enable>;
```

```
wire <output_driver>;
wire <input_signal>;
assign <port> = <output_enable> ? <output_driver> : 1'bz;
assign <input_signal> = <port>;
```

VHDL

Tristate:

```
library ieee;
use ieee.std_logic_1164.all;
entity <tbuf> is
port (
  <enable> : std_logic;
  <input_sig> : in std_logic_vector (1 downto 0);
  <output_sig> : out std_logic_vector (1 downto 0));
end tbuf2;
architecture <port> of <tbuf> is
begin
  <output_sig> <= <input_sig> when <enable> = '1' else "ZZ";
end;
```

Open Drain:

```
library ieee;
use ieee.std_logic_1164.all;
entity <od> is
port (
  <enable> : std_logic;
  <output_sig> : out std_logic_vector (1 downto 0));
end od2;
architecture <port> of <od> is
begin
  <output_sig> <= "00" when <enable> = '1' else "ZZ";
end;
```

Bidirectional:

```
library ieee;
use ieee.std_logic_1164.all;
entity <bidir> is
port (
  <direction> : std_logic;
  <input_sig> : in std_logic_vector (1 downto 0);
  <output_sig> : out std_logic_vector (1 downto 0);
  <bidir_sig> : inout std_logic_vector (1 downto 0));
end bidir2;
architecture <port> of <bidir> is
begin
  <bidir_sig> <= <input_sig> when <direction> = '0' else "ZZ";
  <output_sig> <= <bidir_sig>;
end;
```

Customizing Source Editor

You can set file and window modes, set encoding, and configure line number showing, text wrapping, and printing options in Source Editor.

Setting File and Window Modes

Source Editor offers you the flexibility to configure file and window modes, such as line number showing, text wrapping, and printing options. The configuration can be applied to the current file, the specified types of files, or any new files created in Source Editor.

To set file modes:

1. In Source Editor, choose **View > Language**. You will see list of menu items beside, cpp, hypertext, perl, python, tcl, verilog, vhdl, preference, and edif.
2. Choose the desired file mode from the list. Source Editor will become that specific language sensitive.

To set window modes:

1. In Source Editor, choose **Window > Attach Window**. The Source Editor window is attached to the Diamond main window.
2. If want to detach Source Editor from the Diamond main window, click the Detach Tool icon on the upper-right corner of Source Editor.

Customizing Source Editor from the Diamond Main Window

To customize Source Editor settings in the Diamond main window:

1. In the Diamond main window, choose **Tools > Options**.
2. In the Options dialog, find the Source Editor section.

   You can set settings in the each related section.

   General

   Show Line Number: Displays line numbers of the files (on the left side of the Editor).

   Show Indicator Margin: Displays the side bar where you can click to select the whole row.

   Show Folder Margin: Displays the side bar (on the left side of the Editor) that is used to fold/unfold the syntax.

   Tab Size: Specifies how many spaces are entered when a tab character is pressed.

   Set Default Brings back to the default setting.

   Fonts

   Lists the current fonts used in Source Editor. If you want to change the current fonts, click the font box. From the pop-up Select Font dialog, you can select the desired font. Click **OK** to quit the dialog. The selected font name, font type, and font size will be displayed.

Set Default	Brings back to the default font setting.

Colors

Lists the syntax elements and the current color settings for them. Click the color box to choose a different color.

Set Default	Brings back to the default font setting.

If you want to change to use your own text editor other than Source Editor, you can customize to add your own text editor and associate files to your own text editor program in the Options dialog (**Tools > Options** from the Diamond main window), the **Environment > File Associations** section.

References

This section includes the following references.

Verilog HDL Language Reference

If you have installed Synplify Pro for Lattice, refer to the *Synplify and Synplify Pro for Lattice Reference Manual* in the Synplify Pro installation directory.

VHDL Language Reference

If you have installed Synplify Pro for Lattice, refer to the *Synplify and Synplify Pro for Lattice Reference Manual* in the Synplify Pro installation directory.

Designing with Modules

Modules are functional bits of design that can be re-used wherever that function is needed. Creating such modules with hardware design languages is common practice. To help your design along, Lattice Semiconductor provides a variety of modules for common functions. They are optimized for Lattice device architectures and can be customized. Use these modules to speed your design work and to get the most effective results.

Lattice Semiconductor's modules come in a variety of forms:

- IPexpress provides a variety of functions ranging from the most basic, such as arithmetic and memory, to much more complex functions. With IPexpress these modules can be extensively customized. They can be created as part of a specific project or as a library for multiple projects. See "Creating IPexpress Modules and IP" on page 191.

  However, many of these modules can also be used with PMI (see next item). To decide which method to use, see "Use PMI or IPexpress?" on page 191.

- PMI (Parameterized Module Instantiation) is an alternate way to use some of the modules that come with IPexpress. With PMI, instead of using IPexpress, you directly instantiate a module into your HDL and customize it by setting parameters in the HDL. You may find this easier than using IPexpress if your design requires many variations of the same module. To decide which method to use, see "Use PMI or IPexpress?" on page 191.

- Clarity Designer provides modules similar to those from IPexpress but for ECP5. As with IPexpress, with Clarity Designer you can customize these modules. Clarity Designer also helps you connect these modules to each other and place the PCS and DDR modules in the device's architecture. See "Creating Clarity Designer Modules" on page 211.

- LatticeMico32 microprocessors and LatticeMico8 microcontrollers are exceptions in that they are not customized with IPexpress. LatticeMico32 and LatticeMico8 have their own development environment. To design with LatticeMico System, see "Designing with LatticeMico Platforms" on page 241.

- Reference designs provide you with a starting point on creating your own modules. Lattice Reference Designs are available in Verilog and VHDL, and can be downloaded from the Lattice Web site: www.latticesemi.com/ip.

- Lattice library primitives are very basic functions, such as logic gates and flip-flops. They can be directly instantiated as HDL into designs. But this is an advanced technique and should usually be avoided. For more information, see "Designing with Lattice Library Primitives" on page 209.

Of course you can also create your own modules and that is fully supported too. In fact, Diamond supports creating your own black-box modules. See "Creating Your Own Black Box Modules" on page 206.

Use PMI or IPexpress?

Many IPexpress modules are also available as PMI modules. Why would use one instead of the other?

PMI is a convenient way to use modules of the same type but that vary from instance to instance. This eliminates the need to create a separate module for each instance using IPexpress.

For example, a design might require dozens of FIFOs that are functionally the same but require different address depths. With PMI, you could insert the same FIFO instantiation command wherever it's needed in the HDL and just change the address depth parameter as you go. The alternative would be to use IPexpress to generate different modules for each variation and then insert the instantiation template for each of them. In such a situation, you might find the PMI method easier and faster.

But before deciding to use a PMI module, compare it to the equivalent IPexpress module. Often IPexpress provides more options than PMI does. IPexpress also assures that all of your option selections and parameter settings are legal. PMI offers no error checking.

Also be aware that PMI requires some extra steps for simulation and synthesis. PMI modules are treated as black boxes in design projects. So simulation outside of Diamond, with a third-party simulator, requires compiling a simulation library. And synthesis requires that a synthesis header file be added to the design project or to the project of a standalone synthesis tool, if using one. For more details, see "Requirements for Simulation and Synthesis with PMI" on page 202.

See Also

- List of PMI modules, see Table 144 on page 1402
- "Creating IPexpress Modules and IP" on page 191
- "Using PMI" on page 200

Creating IPexpress Modules and IP

IPexpress is an easy way to use a collection of functional blocks from Lattice Semiconductor. There are two types of functional blocks available through IPexpress: modules and IP. IPexpress enables you to extensively customize these blocks. They can be created as part of a specific project or as a library for multiple projects.

Modules: These basic, configurable blocks come with IPexpress. They provide a variety of functions including I/O, arithmetic, memory, and more. Open IPexpress to see the full list of what's available. Also see the Lattice Module Reference Guide.

IP: Intellectual property (IP) are more complex, configurable blocks. They are accessible through IPexpress, but they do not come with the tool. They must first be downloaded and installed as a separate step before they can be

accessed from IPexpress. To see all that's available and to learn about licensing and other vendors of IP, go to the Lattice Web site: www.latticesemi.com/ip.

Overview of the IPexpress Process Below are the basic steps of using IPexpress modules and IP. For details of performing these steps, see the following topics.

1. Start running IPexpress. It can be started from Diamond's Tools menu after you open your design project. If you want to create a library of configured modules or IP, IPexpress can be opened as a stand-alone tool to create a library of modules.
2. If you want to use a Lattice IP that's not visible, it must be downloaded and installed first. This can be done from IPexpress.
3. Customize the module/IP. These modules and IP can be extensively customized for your design. The options may range from setting the width of a data bus to selecting features in a communications protocol. At a minimum you need to specify the design language to use for the output.
4. Generate the module/IP and bring its .ipx file into your project. Prior to generating the module/IP, select the option "Import IPX to Diamond Project." This will then automatically bring the .ipx file into your project after the generation step completes. If you do not select this option, then after generation, add the .ipx file to your project as you would with any other source file (such as a Verilog or VHDL file). If using IPexpress standalone, there is no project to automatically add the .ipx file.
5. Instantiate the module/IP into the project's design. An HDL instance template is generated during the generation step to simplify this step.
6. IPexpress modules and IP can be further modified or updated later. After the .ipx file has been added to the Diamond project, it is visible in the project's file list. Double-clicking the .ipx file brings up the module/IP's configuration dialog box where changes can be made and the generation process repeated.

See Also

- "Running IPexpress" on page 193
- "Downloading IPexpress IP" on page 193
- "Installing a Downloaded IP" on page 194
- "Generating a Module or IP with IPexpress" on page 195
- "IPexpress Output Files for Modules" on page 197
- "Regenerating a Module or IP with IPexpress" on page 197
- "Importing an IPexpress Module or IP into a Project" on page 198
- "Instantiating an IPexpress Module or IP" on page 199

Running IPexpress

IPexpress can be run as part of the open design project or as a separate, stand-alone tool. When run as part of a project, module selection is limited to modules that work with the project's FPGA. When run as a stand-alone tool, all modules are available and they can be created for any device type. IPexpress is usually run as part of a project. Stand-alone mode can be used to create a library of modules.

On Linux, IPexpress can also be run from the command line as a stand-alone tool.

To run IPexpress with your design project:

- Choose **Tools > IPexpress**.

To run IPexpress as a stand-alone tool, from your Windows desktop:

- From the Start menu, choose **Programs > Lattice Diamond > Accessories > IPexpress**.

To run IPexpress from the Linux command line:

- From the Linux command line, navigate to the <*install_path*>/bin/lin directory and enter **./ipexpress** on the command line.

Note

If this path has been included in your PATH environment variable definition, you do not have to navigate to it.

Downloading IPexpress IP

You can use IPexpress to download and install the latest IP available from Lattice Semiconductor. Before you can download any IP, you need to set up an Internet connection. If you haven't already, choose **Tools > Options**. In the Options dialog box, select **Environment > Network Settings** and fill in the dialog box.

The Web site also has links to other vendors of IP. To see all that's available and to learn about licensing and other vendors of IP, go to the Lattice Web site: www.latticesemi.com/ip.

To download Lattice IP:

1. If you are working on Linux and this is the first IP you have ever installed, go to your home directory and, with a text editor, create an empty file named **ipsetting_l.lst**. Enter no text: the file must be empty and must have this name. If you find you already have a file with this name, do not change it. This is a text file listing your IP and their path names.
2. In IPexpress, click the IP Server button, located at the upper-left of the tool.
3. Click **IP** in the left pane.

The software connects to the Lattice Semiconductor Web site.

4. Expand the folder tree and select the IP you want to download.

   Information about the IP appears in the right pane including links for additional information.

   IP that are not compatible with your version of Diamond have an X on their icons. You can still download and install these IP but you need to upgrade your Diamond software to use them. Look for software requirements in the right pane.

5. Do one of the following:

   - To download and install the IP, right-click the IP and choose **Install** in the pop-up menu.
   - To just download the IP for later installation, right-click the IP and choose **Download** in the pop-up menu.

6. Follow the on-screen instructions.

See Also ▶"Installing a Downloaded IP" on page 194

Installing a Downloaded IP

After downloading an IP, you must install it to make it available in IPexpress.

Installing on Windows **To install the IP:**

1. If IPexpress is open, close it.
2. Go to the location where you saved the IP.
3. Double-click the executable file (.exe).

   The Lattice Semiconductor Setup program opens.

4. Follow the on-screen instructions.
5. Open IPexpress to see the installed IP. Click the Local button and then expand the **IP** folder tree.

   Only IP that work with the selected device and your version of Diamond are available. Unavailable IP have a red X icon, , beside them.

   If you started IPexpress from Diamond's main window, the selection is automatically limited to the device of the design project.

   If you are running IPexpress in stand-alone mode and want access to all the IP, choose **All Device Families** from the menu in the toolbar.

Installing on Linux To install an IP that you downloaded earlier on Linux, the process depends on whether the download is a tar file (.tar.gz) or a shell file (.sh).

To install the IP from a tar file:

1. If this is the first IP you have ever installed, go to your home directory and, with a text editor, create an empty file named **ipsetting_l.lst**. Enter no

text: the file must be empty and must have this name. If you find you already have a file with this name, do not change it. This is a text file listing your IP and their path names.

2. Go to the location where you saved the IP.
3. Enter the following in a command line:

   tar -zxvf <*filename*>**.tar.gz**

4. In IPexpress, click the Import User Configurable IP button.
5. In the dialog box, browse to the installation directory for the IP and select **nodeinfo.cfg**.
6. Click **Open**.
7. To see the installed IP, click the Local button and then expand the **IP** folder tree.

To install the IP from a shell file:

1. Go to the location where you saved the IP.
2. Enter the following in a command line:

   sh <*filename*>**.sh** <*Diamond install path*>**/bin/lin** [<*destination path*>]

   The <*destination path*> is the full path name of where you want the IP to be installed. If not included, the IP will be installed in the current directory.

   The first time you install an IP, IPexpress creates a file, ipsetting_l.lst, in your home directory. This is a text file listing your IP and their path names.

3. To see the installed IP, click the Local button and then expand the **IP** folder tree.

Generating a Module or IP with IPexpress

Use IPexpress to generate a customized functional block from any module or installed IP.

Note

Lattice Synthesis Engine (LSE) is not currently used to generate IP. When LSE is selected to be used for your design, the IP generation step (done within IPexpress) will use Synplify Pro.

ROM modules require that a memory initialization (.mem) file be available before generating a customized module. The memory initialization file is optional with RAM modules. See “Creating a Memory Initialization File with Memory Generator” on page 203.

To generate an IP or module:

1. If you are running IPexpress as a stand-alone tool, then choose a device family from the menu in the toolbar. Only modules and IP that work with

that device and your version of Diamond will be available. Unavailable modules will have a red X icon,  or  , beside them.

If you started IPexpress from a Diamond project (Diamond's main window), the functional block selection is automatically limited to the device of the design project.

2. In the Module/IP tree, select the module or IP that you want to generate.
3. To get more information about the module or IP, click the **About** tab.
4. In the Configuration tab, specify general project information and the base file name for the module or IP. If you started IPexpress from a Diamond project (Diamond's main window), all but File Name and Module Output have been filled in from the design project. You can still change Project Path if you want.
   - Project Path is the location for the customized module's files.
   - File Name is the base name for the module's files (that is, with no extension).
   - Module Output is how the module will be coded.
   - Device Family is the device family that the module will be optimized for. This can be different from the device family selected in the toolbar.
   - Part Name is the exact device that the module will be optimized for.
   - Synthesis is the synthesis tool that will be used while generating the module.
5. Click **Customize.**

   The module's dialog box opens.
6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information.

   As the options change, the schematic diagram of the module changes to show the ports and the device resources the module will need.
7. To automatically import the .ipx file into your project when the module/IP is generated, select **Import IPX to Diamond project** (not applicable in IPexpress stand-alone mode). If you do not select this option, the .ipx file must be added to the project. See "Importing an IPexpress Module or IP into a Project" on page 198.
8. Click **Generate**.
9. Click the **Generate Log** tab to check for warnings and error messages.

See Also

- "Creating a Memory Initialization File with Memory Generator" on page 203
- "IPexpress Output Files for Modules" on page 197

IPexpress Output Files for Modules

IPexpress creates the following output files for modules under the specified Project Path. The <*file_name*> comes from the File Name specified in the Configuration tab.

IPexpress creates some different files for IP. These are documented in the IP's associated user guide.

Table 3:

File Name	Description
<*file_name*>.ipx	Manifest file. This file is loaded into IPexpress so that modifications can be made to the module.
<*file_name*>.lpc	Parameter file.
<*file_name*>.naf	Internal file. Do not edit.
<*file_name*>.srp	Report file.
<*file_name*>.sym	Internal file. Do not edit.
<*file_name*>.tmpl.v	Instantiation template for Verilog netlist.
<*file_name*>.tmpl.vhd	Component/instantiation template for VHDL netlist.
<*file_name*>.txt	Configuration file for PCS and System Bus modules.
<*file_name*>.v	Verilog HDL file for both synthesis and simulation. Verilog output files declare implicit wire types.
<*file_name*>.vhd	VHDL file for both synthesis and simulation.
<*file_name*>_generate.log	IPexpress log file.
msg_file.log	Log message file.
tb_<*file_name*>_tmpl.v	Verilog test bench template.
tb_<*file_name*>_tmpl.vhd	VHDL test bench template.

Regenerating a Module or IP with IPexpress

By regenerating a customized module or IP you can modify any of its settings including: device type, design language, and any of the options specific to the module. You can also update older modules or IP to the latest version.

Regenerating can be done to modify an existing module or to create a new but similar one.

Note

If the module was originally customized with ispLEVER, you may see an .lpc file but no .ipx file. The .ipx file is new with Diamond. However, Diamond also supports .lpc files for backward compatibility. When you regenerate a module from ispLEVER, the .lpc file is replaced with an .ipx file.

To just change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 below.

To regenerate a module or IP:

1. In IPexpress, click the Regenerate button.
2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.
3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the Target box.
4. If you want to generate a new set of files in a new location, set the location in the IPX Target File box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.
5. Click **ReGenerate**.

   The module's dialog box opens showing the current option settings.
6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information.

   As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project, if it's not already there, select **Import IPX to Diamond project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the **Generate Log** tab to check for warnings and error messages.
10. Click **Close**.

Importing an IPexpress Module or IP into a Project

After generating module source files using IPexpress, you can import the module by importing the IPexpress manifest file (.ipx). Modules and IP have several files of different types but you only need to import the .ipx. The .ipx file identifies the components needed to make up the module or IP.

Note

If the module was originally customized with ispLEVER, you may see an .lpc file but no .ipx file. The .ipx file is new with Diamond. However, Diamond also supports .lpc files for backward compatibility. If there's no .ipx file, select the .lpc file.

Importing the module may not be necessary if the "Import IPX to Diamond project" option was selected when the module was generated.

To import a module:

1. In the File List view, right-click the implementation folder () and choose **Add > Existing File**.
2. Browse for the customized module's .ipx file, <*file_name*>**.ipx**, and select it.
3. Click **Open**.

   The .ipx file is added to the File List view.
4. Check the Output Panel for error messages. If the module is not targeted for the current device, try double-clicking the file to regenerate the module.

After importing the module, you can further customize it for your design project including changing the device type, design language, and any of the options specific to the module. You can also update older modules or IP to the latest version. See "Regenerating a Module or IP with IPexpress" on page 197.

Instantiating an IPexpress Module or IP

IPexpress modules and IP are instantiated the same way other modules are in your HDL. When you generate modules and IP in IPexpress, the tool also produces a Verilog or VHDL file with the necessary instantiation commands. You can copy and paste the contents of this file into one of your source files.

Before you instantiate any IPexpress module, check that it is compatible with your design project's device. When they were created, the modules were optimized for a specific device and may depend on that device's architecture. A module may not work with a different type of device and may need to be regenerated. If so, see "Regenerating a Module or IP with IPexpress" on page 197.

The instantiation file is located in the folder that the module was created in. The file name is based on the module's name: <*module name*>_tmpl.v or <*module name*>_tmpl.vhd.

Copy the instantiation command and paste it into one of your source files. Then add an instance name and signal names to the module ports.

Note

The following I/O modules from IPexpress cannot be used as the top level of a design. These must be instantiated within another module: DDR, DDR_GENERIC, DDR_MEM, DQS, and SDR.

Before running any processes on the design, make sure that the IPexpress module or IP has been imported into the project. See "Importing an IPexpress Module or IP into a Project" on page 198.

Using PMI

PMI (Parameterized Module Instantiation) is an alternate way to use some of the modules that come with IPexpress. With PMI, instead of using IPexpress, you directly instantiate a module into your HDL and customize it by setting parameters in the HDL. You may find this easier than using IPexpress if your design requires many variations of the same module.

But PMI requires some extra steps for simulation and synthesis. PMI modules are treated as black boxes in design projects. So simulation outside of Diamond, with a third-party simulator, requires compiling a simulation library. And synthesis requires that a synthesis header file be added to the design project or to the project of a standalone synthesis tool, if using one.

See Also

- “Use PMI or IPexpress?” on page 191
- List of PMI modules, see Table 144 on page 1402
- “PMI Process Steps” on page 200
- “Instantiating a PMI Module” on page 201
- “Requirements for Simulation and Synthesis with PMI” on page 202

PMI Process Steps

This section provides a brief listing of the process steps you take when using PMI.

1. Using Source Editor, insert PMI modules in the HDL source files. This is the recommended method because it includes the PMI instance templates, which require minimal editing.
2. Observe the PMI requirements to ensure successful functional simulation.

   Since PMI block generation occurs post-synthesis, functional simulation uses simulation models for these blocks. If you are running simulation in Aldec Active-HDL, these models are automatically used. If you are using another simulator, you must first compile the simulation library.

   See “Requirements for Simulation and Synthesis with PMI” on page 202 for more details.
3. Add the PMI synthesis header files to your design project or to your stand-alone synthesis tool’s project.

   When synthesis is run, your PMI block instance will pass through the process as a black box definition. The PMI synthesis header file contains a comprehensive list of all necessary module/component declarations. This saves you from having to create these declarations yourself or to obtain them from some other source.

   See “Requirements for Simulation and Synthesis with PMI” on page 202 for details.
4. After synthesis, run the Translate Design process (EDIF2NGD and NGDBUILD). Note: If you are using LSE as the synthesis tool, Translate

Design is automatically included with synthesis. During this step, the module generation that is required to implement the various instances in your design is performed. This process will result in the output of multiple .ngo files that are then input into one output .ngd file.

5. Perform the remaining implementation steps as you would normally.

Instantiating a PMI Module

PMI modules are instantiated the same way other modules are in your HDL. The Diamond software provides a template for the Verilog or VHDL instantiation command that specifies the customized module's ports and parameters. You customize the module by changing the parameters.

To instantiate a PMI module:

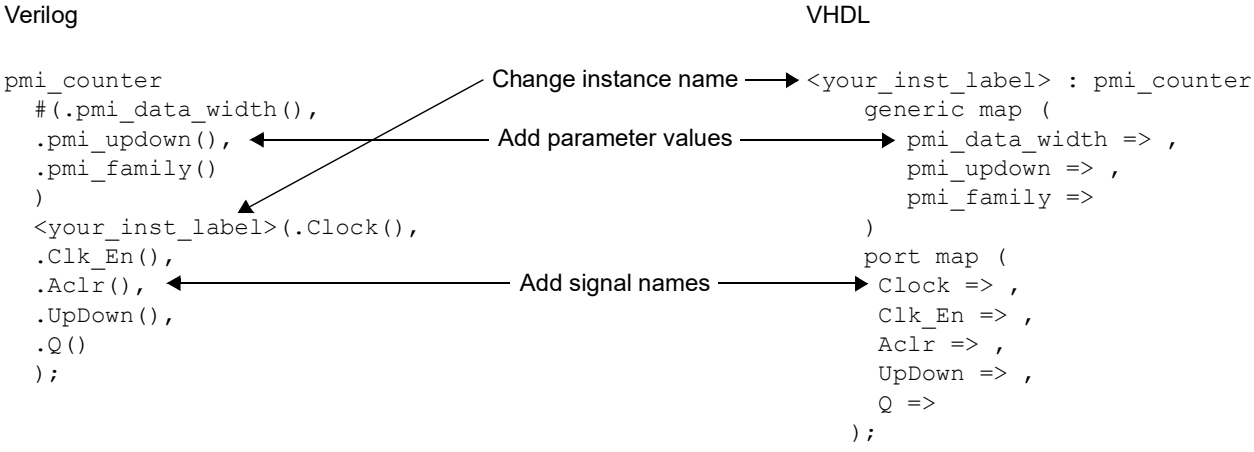
1. With Source Editor, open the source file that will receive the module. See "Running Source Editor" on page 167.
2. Place the cursor in the desired spot for the instantiation command.
3. Choose **View > Template Editor**.

   The Template Editor view appears.
4. In Template Editor, open the **System Templates > PMI** folder and select the module.

   The code of the template shows in the lower box of the Template Editor view.
5. Right-click the template and choose **Insert to text**.

   The instantiation command is copied into your source file.
6. Add an instance name, set parameter values, and add signal names to the corresponding module ports in the instantiation command. For information about allowed parameter values, see the section on the module in the "Lattice Module Reference Guide" on page 1400.

Figure 20: PMI Instantiation Command Example



7. Save and close the source file.

Note

Simulation and synthesis require additional preparation. VHDL also has some additional requirements. See "Requirements for Simulation and Synthesis with PMI" on page 202.

Requirements for Simulation and Synthesis with PMI

Unlike IPexpress module generation, which produces source files that contain all of the necessary code for simulation, the PMI module code is a black box definition of an IPexpress module. For this reason, simulation models for these blocks must be available to supplement the code for simulation. For synthesis, a PMI synthesis header file needs to be added to the project.

VHDL designs have a few additional requirements for both synthesis and simulation.

PMI Simulation Models For Aldec Active-HDL, the PMI models are automatically available. When using ModelSim, or any other simulator, you must first compile the simulation library (pmi_work) from the sources, which are in:

<*install_dir*>/cae_library/simulation/verilog/pmi

Note

No VHDL PMI models exist. The pmi_work library is compiled from Verilog modules and used in both Verilog and VHDL simulation. The VHDL simulation must run in a mixed-HDL environment.

PMI Synthesis Header Files The PMI synthesis header files for Verilog and VHDL, respectively, are at:

<*install_dir*>/cae_library/synthesis/verilog/pmi_def.v
<*install_dir*>/cae_library/synthesis/vhdl/pmi_def.vhd

If you are running synthesis from inside Diamond (the integrated flow), add the PMI synthesis header file to your design project by right-clicking the implementation folder () in the File List view and choosing **Add > Existing Source**.

If you are running the synthesis tool outside Diamond (the interactive or stand-alone flows), add the PMI synthesis header file to your synthesis tool project.

VHDL Requirements When performing simulation for VHDL PMI, you also need to compile the PMI component declaration file into your work library. The file is at:

<*install_dir*>/cae_library/simulation/vhdl/pmi/pmi_def.vhd

At the top of your source file, add two lines:

```
use work.pmi_components.all;
library pmi_work;
```

The first line includes the PMI components defined in the header file. The second line is required by some simulators, such as Active-HDL, for successful library binding.

Creating a Memory Initialization File

A memory initialization (.mem) file is an ASCII text file that specifies the initial contents for the memory modules in your design. A .mem file is required to create ROM and optional when creating RAM modules with IPexpress or PMI.

A .mem file can be most easily created or edited using Diamond's Memory Generator. But you can also create a .mem file using any text editor. Use the file format described in the following section.

See Also

- "Creating a Memory Initialization File with Memory Generator" on page 203
- "Memory Initialization File Format" on page 204

Creating a Memory Initialization File with Memory Generator

Diamond's Memory Generator is the easiest way to create or edit a memory initialization file.

To create a memory initialization file:

1. Choose **File > New > File**.

   The New File dialog box opens.
2. Under Categories, click **Other Files**.
3. Under Source Files, click **Memory Files**.
4. Fill in a name and location for the new memory initialization file.
5. Click **New**.

   The Memory Generator tab opens showing the file name that you specified.
6. In the Options view, set file format and memory size in the File Format, Depth, and Width boxes. See "Memory Initialization File Format" on

page 204 for more information. Then click **Update** to apply your changes and refresh the display.

Note

If you close the Options view and want to get it back, choose **View > Options View**.

7. In the Options view, choose the Address Radix and Data Radix for the display.

Note

The Radix settings only specify in which format the addresses and data values are displayed and entered in the Memory Generator window. They do not affect the memory data format recorded in the .mem file. The data format in the .mem file depends on the File Format option.

8. Fill in the memory contents in the right pane using the following methods:
   - Click in a cell and type in the value. You can use the arrow and Tab keys to move between cells.

     The values you enter should match the format specified in the Data Radix box. Values that are too large are displayed in blue. You are not allowed to save a file with blue values.
   - To find a location in a large memory, type the address into the “Jump to” box and click **Go**.
   - To fill a block of memory with the same value, type the first and last addresses of the block into the From and To boxes (under Filling). Type the memory value into the Fill box. Then click **Fill**.
9. When you finish, choose **File > Save** <*filename*>.

See Also ▶“Memory Initialization File Format” on page 204

Memory Initialization File Format

A memory initialization (.mem) file is an ASCII text file that consists of a header followed by lines of memory data. The data must be in one of the following formats: Bin (binary), Hex (hexadecimal), or Address-Hex (described below).

For hexadecimal values, both upper and lower case can be used. If the data has fewer bits than the specified data width, the most significant bits are filled with 0. Any address not specified will be filled with 0.

Comments can be added at any point after the header (defined below) by starting the comment with a pound sign (#) or two slashes (//). The comment then includes everything to the end of the line. Comments may be added to any of the data, but never add comments to the header.

Header A .mem file starts with a header, which declares the file format, memory size, and address and data display radix for Memory Generator. The syntax of the header is:

#Format=Bin | **Hex** | **AddrHex**
#Depth=<*1-65536*>
#Width=<*1-256*>
#AddrRadix=<*index_number*>
#DataRadix=<*index_number*>
#Data

The <*index_number*> can be one of the following numbers. AddrRadix and DataRadix can have different values.

Binary: 0
Octal: 1
Decimal: 2
Hexadecimal: 3

For example, the following header says the .mem file is using the binary format for a 32x8 memory. When displayed in Memory Generator, the address will be shown in hexadecimal and the data will be shown in binary.

```
#Format=Bin
#Depth=32
#Width=8
#AddrRadix=3
#DataRadix=0
#Data
```

Bin and Hex Formats The data is represented in binary or hexadecimal format. Each line of data specifies the contents for one memory location, starting with address 0. That is, the first line is for address 0, the second line is for address 1, and so on. For each line, the data is interpreted as least significant bit on the right.

For example, in the Bin format, the following lines will initialize address 0 to “00011011”, address 1 to “11111010” (assuming it is a 32x8 memory).

```
# for a 32x8 memory
11011
11111010
```

In the Hex format, the following lines will initialize address 0 to “003B”, address 1 to “FB0A” (assuming it is a 32x16 memory).

```
# for a 32x16 memory
3B
FB0A
```

AddrHex The data is represented in hexadecimal format. Each line consists of an address followed by a colon and then any number of data words, separated by spaces:

<*address*>:<*data*> <*data*> <*data*> ...

The data will be applied starting at <*address*> and filling in sequentially from there.

For example:

```
A0:03 F3 3E 4F
B2:3B 9F
```

will initialize A0 with 03, A1 with F3, A2 with 3E, A3 with 4F, B2 with 3B, and B3 with 9F. The other addresses will be initialized to 0. So A4 through B1 will be set to 0.

See Also ▶"Creating a Memory Initialization File with Memory Generator" on page 203

Creating Your Own Black Box Modules

In some cases, you may not want to distribute HDL source code because of the risk of changes or of exposing proprietary information. So, Lattice Semiconductor offers a compiled Native Generic Object (NGO) netlist format as an alternative to HDL.

Advantages and Disadvantages An NGO netlist has the following advantages over HDL source code:

- Hides details of internal logic
- Easy to distribute
- Optimized to meet timing or area requirements
- Optional grouping and floorplan constraints

On the other hand, an NGO netlist:

- May not be portable across all device families
- Cannot be parameterized

Overview of the Black Box Process The following are the basic steps for creating and using a black box module. For details about performing these steps, see the topics listed under "See Also."

1. Create an NGO netlist.

   Start with a design project just for the module and add some attributes. Then run the synthesis and Translate Design processes.

2. Create support files.

In addition to the NGO netlist, users of the black box module will need additional information such as declaration and instantiation templates, a data sheet, a simulation model, and timing attributes.

Note

NGO blocks will be given a unique name which is constructed by appending the parameter value to the name of the NGO module as in the following example.

For the following module :

```
module my_add_sub (DataA, DataB, Add_Sub, Result);
parameter bit_width = 6;
...
endmodule
```

After generating the NGO file, the ngo must be renamed with the parameter value appended to the module name as shown below:

```
"my_add_sub_6.ngo"
```

3. Instantiate the module.

   Following the instructions from the module's data sheet, copy the declaration and instantiation templates into your design project.

See Also

- "Creating an NGO Netlist" on page 207
- "Support Files for the Black Box Module" on page 208
- "Instantiating a Black Box Module" on page 209

Creating an NGO Netlist

To create an NGO netlist for a black box module, you just need to add synthesis attributes to the ports and run the synthesis and Translate Design processes. The exact procedure depends on your synthesis tool: Translate Design runs as a part of Lattice Synthesis Engine (LSE); Translate Design is a separate step with Synplify Pro.

To create an NGO netlist for a module:

1. Create a design project for the module.
2. If using:
   - LSE, set "Use IO Insertion" to False in the Synthesis options.
   - Synplify Pro, set "Disable IO Insertion" to True in the Synthesis options.
3. If you want to include floorplan information, add grouping (UGROUP/HGROUP) or placement (REGION, LOC) attributes.

   Floorplan constraints can improve performance with some designs but may make the NGO netlist less portable across devices.
4. Synthesize the design.

If using LSE, an NGO netlist (.ngo) file is created.

5. If using Synplify Pro, double-click **Translate Design** in the Process view.

   An NGO netlist (.ngo) file is created.

Support Files for the Black Box Module

To maximize the value of a black box module, you need to create additional material such as declaration and instantiation templates, a data sheet, a simulation model, and timing attributes. The declaration and instantiation templates are the minimum information someone else will need to use the module. The other items are recommended but not required.

Declaration Templates Users will need to know how to declare the module in their designs. The simplest way to is to supply a file with the code. In the file, create an empty Verilog module or VHDL architecture declaration. Add “black box” synthesis attributes for the module and any ports that must attach to external PIOs. See Figure 21 on page 208 for an example (the synthesis attribute is for Synplify Pro).

Figure 21: Verilog Sample Declaration

```
module top (DAT_B, DAT_A, CD, CK, DAT, OE)
/* synthesis syn_black_box black_box_pad_pin="DAT_B[15:0]" */;
input CD;
input CK;
input OE;
input [15:0] DAT;
inout [15:0] DAT_B;
output [15:0] DAT_A;
endmodule
```

Instantiation Template Users will also need a template for instantiating the module. In another file, create a Verilog or VHDL instantiation template such as shown in Figure 22 on page 208. Include the “FILE=” attribute for the .ngo file name. The synthesis attribute is for Synplify Pro.

Figure 22: Verilog Sample Instantiation Template

```
// Register/Bidirectional Circuit (Lattice NGO)
top top_1_ (
.DAT_B (DAT_B),
.DAT_A (DAT_A),
.CD (CD),
.CK (CK),
.DAT (cnt),
.OE (OE))
/* synthesis FILE="top.ngo" */;
```

Data Sheet Users will probably need more information to use the module effectively. So create a data sheet for the module with details about what the module does, I/O port function and format, and support contacts.

Simulation Model To fully simulate their designs, users will need a simulation model of the module. Base the model on RTL code. If there are restrictions on how the module is used, pre-compile the model for the target simulator. If you provide the RTL itself, make it clear that the RTL should not be re-synthesized unless you are willing to risk timing changes.

Timing Attributes To fully analyze the timing of their designs, users will need timing information for the module. Create an example use of the module and run static timing analysis. Then add “black box timing” attributes to the Verilog module or VHDL component declaration file.

Instantiating a Black Box Module

A black box module may come with a few files besides the .ngo file. The module should come with a template for the instantiation command and an empty Verilog module or VHDL component declaration. The module may also come with a datasheet, a simulation model, or an example implementation. See “Support Files for the Black Box Module” on page 208.

If there’s a datasheet, refer to it for information about the module and instructions for its use.

Note

Check what device the module was created for. If your design project is not using the same or a very similar device, the module may not work.

To instantiate an NGO module:

1. Do one of the following:
   - Copy the .ngo file to the local project folder.
   - Specify the file’s location in the Macro Search Path setting of the Translate Design process in the project’s strategy.
2. Copy and paste the instantiation template and the module declaration into your design files. This can be done with Diamond’s Source Editor or you can use any text or HDL editor.
3. In the instantiation command, add an instance name and signal names to the corresponding module ports.

Designing with Lattice Library Primitives

Any Lattice library primitive described in the “FPGA Libraries Reference Guide” on page 1665 can be instantiated as a Verilog module or VHDL component in your RTL design. This sort of “gate-level” design can be error-prone and should be limited to a small number of primitives if attempted at all. In general, Lattice recommends you rely on IPexpress to generate modules that are built with Lattice library primitives.

To minimize the amount of code overhead required to design with a library primitive, Lattice provides a Verilog and VHDL synthesis header library file for each major FPGA device family. Refer to the “Lattice Synthesis Header Libraries” on page 579 topic for details. Typically the module is treated as a “black box” which causes the synthesis tool to pass instances of the library primitive into the target netlist untouched.

Global signals for global set/reset (GSR), power-up reset (PUR), tri-state all (TSALL), and the internal oscillator (OSCA, OSCC, OSCD, OSCE, OSCF) can be used within structural models built with Lattice library primitives. For more information, see “How to Use the Global Set/Reset (GSR) Signal” on page 1174, “How to Use the Tristate Interface (TSALL) Global Signal” on page 1190, and “How to Use the Internal Oscillator” on page 1192

The “FPGA Libraries Reference Guide” on page 1665 contains descriptions, pinouts, and schematic diagrams of all library primitives for Lattice FPGA libraries.

See Also ▶“FPGA Libraries Reference Guide” on page 1665

- “Lattice Synthesis Header Libraries” on page 579
- “Simulation Library Files” on page 317
- “How to Use the Global Set/Reset (GSR) Signal” on page 1174
- “How to Use the Internal Oscillator” on page 1192
- “How to Use the Power Up Set/Reset (PUR) Global Signal” on page 1189
- “How to Use the Tristate Interface (TSALL) Global Signal” on page 1190

Creating Clarity Designer Modules

Improving design entry has been a goal for many generations of software tools. Early tools used only designer entered source and very small basic cell library blocks. Later tools expanded the area for pre-made or generated blocks. But these tools only provide single-block generation. As multiple complex blocks are used in devices, resource allocation problems often occur. These issues require blocks to be planned together and potentially can even require blocks to be generated with certain options to allow maximum utilization. To solve these problems tools that can provide capabilities beyond simple module generation are required.

Clarity Designer Benefits and Features

Clarity Designer is a tool within the Lattice Diamond software environment that addresses the need to be able to generate and plan multiple blocks together. Clarity Designer is used for configuration of blocks, building the connections between blocks, and planning the resources used by the PCS and DDR blocks in the design. For device families supported by Clarity Designer, IPexpress functionality is accomplished along with functionality for building and planning. The IPexpress tool is disabled when using a device family supported by Clarity Designer. Device families that are not supported yet by Clarity Designer still require the use of IPexpress. Clarity Designer is currently only available for the ECP5 device family. A comparison chart between IPexpress and Clarity Designer features is shown in Table 4.

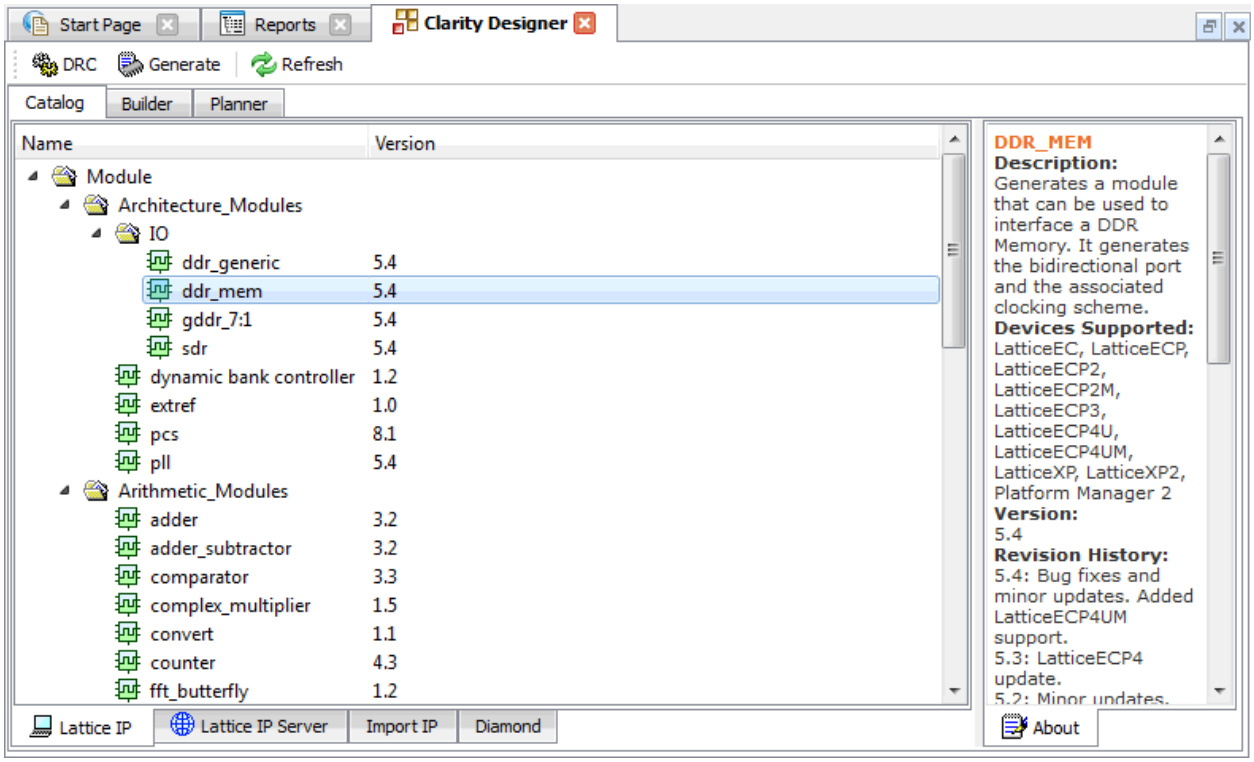
Table 4: IPexpress versus Clarity Designer

		IPExpress	Clarity Designer
Configuration / Generation			
	Modules	Yes	Yes
	IP	Yes	Yes
	Download IP	Yes	Yes
Building			
	Rule checking	No	Yes
	Generate Connectivity	No	Yes
	Connection Assistance	No	Yes
	Design Reuse	No	Yes
Planning (PCS & DDR)			
	Pre-Synthesis	No	Yes
	Placement Assistance	No	Yes
	Rule checking	No	Yes
	Graphical usage	No	Yes

Configuration and Importing

Clarity Designer produces design (.sbx) files that can be used in the design project like an HDL file or an IPexpress generated (.ipx) file. A key difference compared to IPexpress generated files is that Clarity Designer files can contain multiple modules or IP blocks, not just a single block and can represent a subsystem. With IPexpress, the process is to generate a module or IP. This is one step since an IPexpress file can only contain one module or IP. For Clarity Designer, the saving of a file is a separate step. Modules or IP are configured and then more modules or IP can optionally be added within the same file. Additionally, since building and planning can also be done, saving the file and generating the blocks needs to be done as a later step.

Figure 23: Clarity Designer Module and IP Configuration



Configuring modules is done by going to the **Catalog** tab, selecting the **Lattice IP** tab at the bottom of the window, and selecting a module or IP and then double-clicking. A separate dialog box will open that allows the appropriate options to be set and, when the configure button is pressed, this information will be added to the Clarity Designer file that is currently open.

To add new IP, when the **Catalog** tab is selected, select the **Lattice IP Server** tab at the bottom of the window, and click the IP folder to get an updated list of the available IP. Select the IP you want to download from the available list and click the **Download** or **Install** button. Once the IP is installed, it can be configured as explained above.

If an existing project is being migrated to Clarity Designer from IPexpress, existing IPexpress files in the design must be imported and saved as Clarity Designer files. To import existing .ipx files, in the **Catalog** tab, select the

Import IP tab at the bottom of the window. In the window you can now select the existing file and include the name of the converted module for use in Clarity Designer. After completing the necessary fields, press the Import button to complete the process. Each existing IPexpress file must be imported individually.

A unique feature in Clarity Designer compared to IPexpress is the ability to use an RTL module within the design as a block within the Clarity Designer file. This is the same as instantiating a block within an HDL module. It is important to understand that RTL modules are imported into Clarity Designer are not automatically updated when the RTL is changed. The tool must be manually refreshed if the RTL is updated. To import an RTL module, the HDL source code must already be in the Diamond project and showing in the file list. Then from the **Catalog** tab, select the **Diamond** tab at the bottom of the window. This will show an elaborated view of the design hierarchy. Double-click the instance that should be imported. The imported module will be available in the Builder tab for connecting to the rest of the design.

Building

Once one or more modules or IP have been configured, then any connections between them or to the top of the subsystem (file) can be set. Each Clarity Designer file is the equivalent of a subsystem like an HDL module. Information in the file can be self-contained within the subsystem or connected to the top where it can connect to a higher level of the design hierarchy. The Builder function allows blocks to be connected to each other or to the top of the design file (equivalent to an HDL module description of the blocks and connections). The Builder tab provides two ways to approach this task. See Figure 24 on page 214.

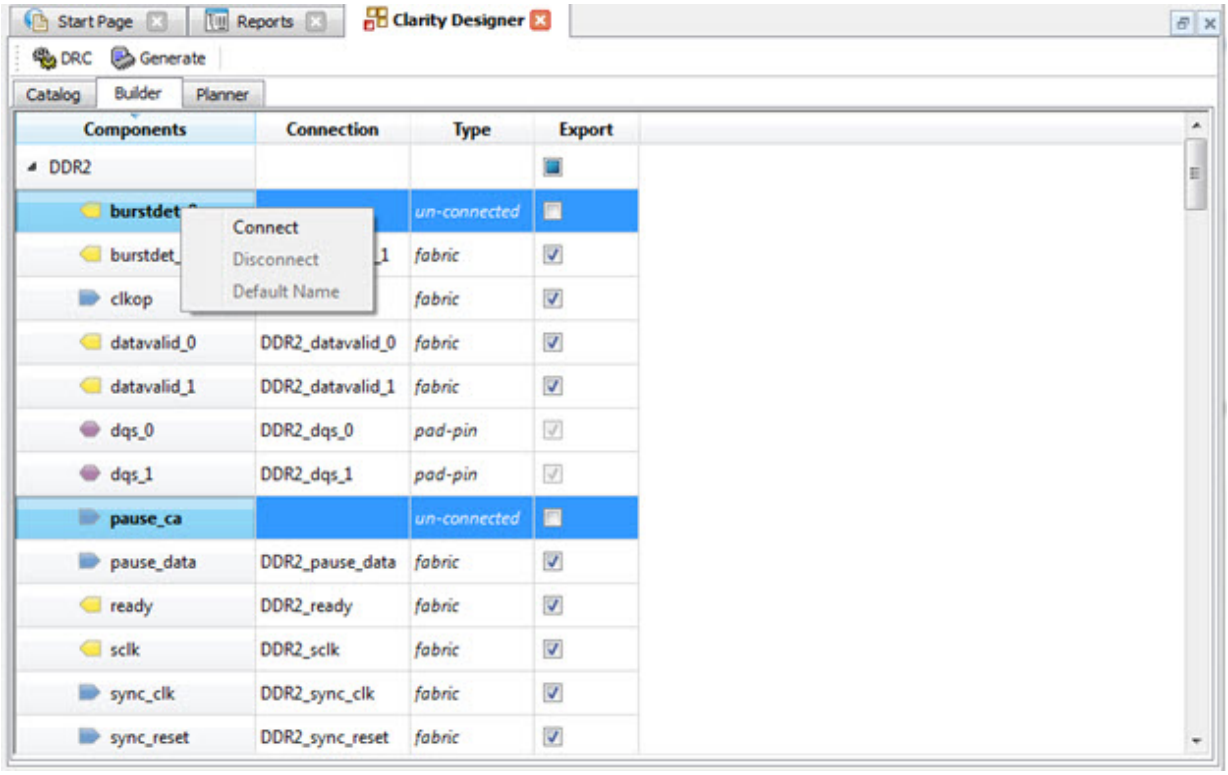
In the Components view, there are columns showing the components used, connections, connection type, and a column named “Export.” The Export column is for connecting a signal to the top of the Clarity Design file. Checking this column for a signal means that signal connects to the top of the file so it can be used at a higher level in the hierarchy. By default the Export column is checked when a module is created in order to mimic the behavior with IPexpress for creating a file with a single module. The Export column should be unchecked for signals that are connections between modules.

The Schematic view provides a schematic of the Clarity Designer module including a block for each module that you’ve added, external ports, and wires connecting them all.

Ports can be connected to each other or to the top of the Clarity Designer module in one of two ways in the Schematic view:

- Selecting them and then choosing from the right-click menu. To connect signals between modules, select the first signal, then Control-click on the second signal. Right-click on either signal name and then select **Connect** from the popup menu to complete the connection. Multiple signals can be connected by selecting them in this manner. To disconnect signal connections between modules, right-click on the signal and select **Disconnect** in the popup menu. Signals can be renamed by double-

Figure 24: Clarity Designer Building



clicking the signal name and typing in the new name. Refer to "Building with the Schematic View" on page 226.

- Using right-click and drag- and-drop functionality to connect available components, ports, and pins to other available components, ports, and pins. Refer to "Building with the Schematic View" on page 226.

The primary use of the builder functionality is to improve ease of use for connecting multiple modules. To connect signals between modules, select the first signal, then Control-click on the second signal. Right-click on either signal name and then select **Connect** from the popup menu to complete the connection. Multiple signals can be connected by selecting them in this manner. To disconnect signal connections between modules, right-click on the signal and select **Disconnect** in the popup menu. Signals can be renamed in the Property and Design tabs by double-clicking the signal name and typing in the new name. Signals cannot be renamed in the Schematic View.

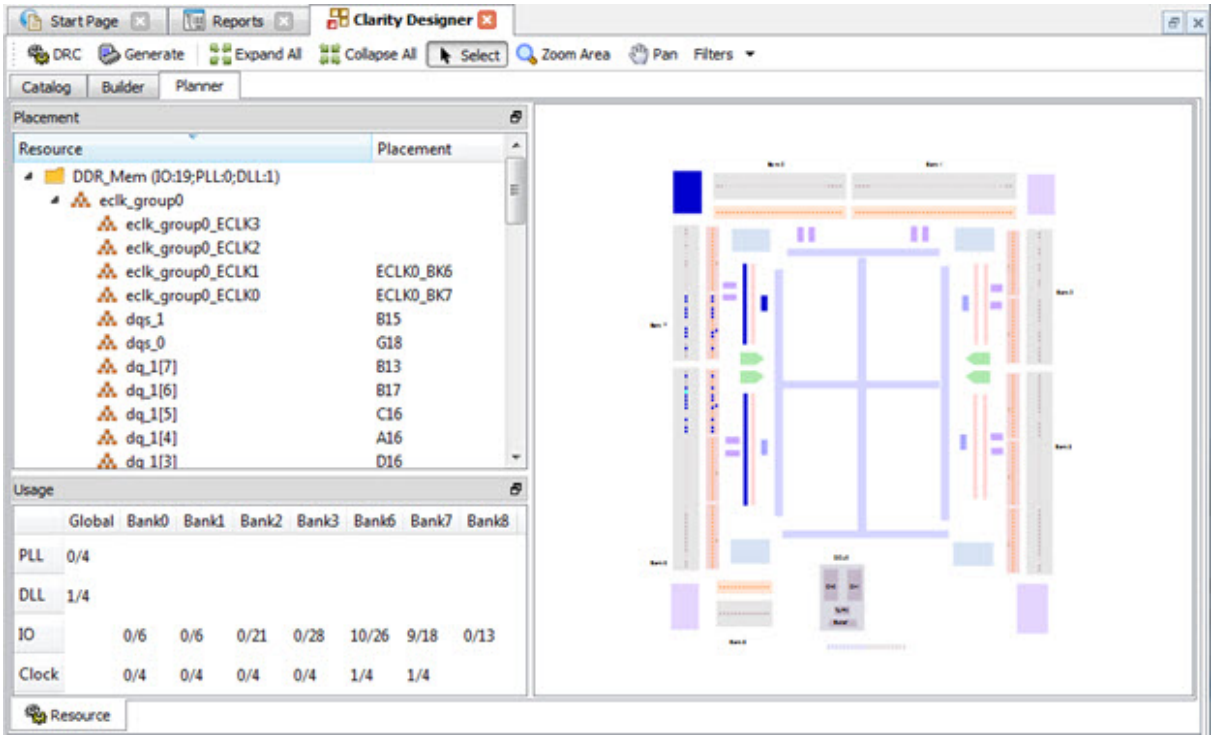
Planning and Determining Usage

After configuring and connecting modules, modules and IP can be placed in the Planner tab. Currently only PCS-based modules and IP, and DDR-based modules and IP are supported for placement. Other design elements should still be placed using Spreadsheet View or through LPF constraints.

In addition, the real-time resource usage report allows you to monitor the resource usage during the Planning phase. The usage information is updated

after each planning operation. The report displays used resources and total available resources, as shown in Figure 25.

Figure 25: Clarity Designer Planning



Design elements that can be placed are shown in the left panel and a chip-level view of the FPGA and available resources are shown on the right. Elements can be dragged from the left panel onto the appropriate resource on the right panel to be placed. When an element is dragged onto a resource that can be used, the cursor will change to a “+” to show a valid solution is possible. The chip view on the right can be zoomed in and out to help with placing the design elements. Design elements can be dragged as a group or as individual elements to be placed.

For details on how to do placement on PCS/SERDES and DDR modules, “PCS/SERDES Planning” on page 227 and “DDR I/O Planning” on page 233.

Diamond Integration

Clarity Designer outputs source files that are used in a Diamond project like other source file types such as VHDL, Verilog, EDIF, or IPexpress files. Clarity Designer files must be added to the file list in order to be used for a project.

To be added to the file list, the file must be generated. This should be done after the configuration, building, and planning steps. To generate the file, click the Generate button in the Clarity Designer window toolbar.

Currently Diamond projects support using Clarity Designer files either as a module for a level of hierarchy within a design project using HDL as the top level or as the top level of the design project.

Usage 1—Used as a Module If a Clarity Designer file is used as a module within an existing design, it must be instantiated within an HDL file, similar to how IPexpress modules are instantiated within an existing HDL design. Within an HDL design, multiple Clarity Designer files can be used. However, within a Clarity Designer file, an instantiated module cannot be another Clarity Designer file.

Currently, using Clarity Designer files as modules within an HDL design is recommended. This allows the current hierarchy of a design to be used without significant changes. If a Clarity Designer file is used as the top-level of the design, the design hierarchy must meet the requirements specified for top-level use.

Note that when instantiating generated subsystems in an HDL module, it is important to use the instance template generated by Clarity Designer. Top-level I/Os in the subsystem use port names based on instance and port names by default. Additionally, the instance needs an HDL attribute to indicate it is a Clarity Designer instantiation. Templates can be found in the directory in which the .sbx file was saved.

Usage 2—Top-Level of Design If a Clarity Designer file is used as the top-level of the design, the following requirements must be followed.

- All modules and IP must be directly instantiated within this file. Modules cannot be instantiated within an HDL file. For example, a PLL module must be in the top-level Clarity file. It cannot be instantiated within an HDL file.
- Any HDL in the design must be imported into the file as a module. HDL files can contain a hierarchy of other HDL modules, but no Clarity Designer modules or IP can be instantiated within the HDL.
- Clarity Designer files cannot be used as modules within another Clarity Designer file.

Basic Tasks

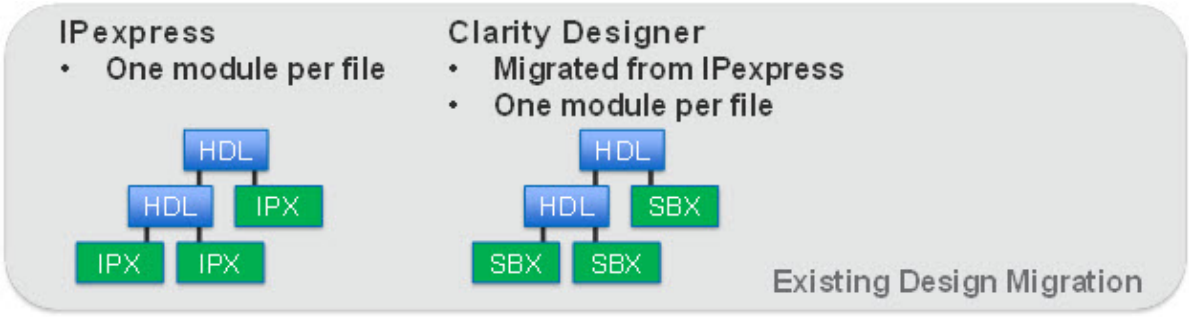
Tool usage falls into two primary methods: migrating existing designs in Clarity Designer and starting new designs in the tool.

Design Migration versus New Designs

When migrating an existing design into Clarity Designer, it is recommended that the design hierarchy should be unchanged and each existing IPexpress (.ipx) file should be imported into Clarity Designer and replaced with a new Clarity Designer (.sbx) file as illustrated in Figure 26 on page 217. The exceptions to this are PCS/SERDES and DDR modules and IP. Since these should be placed together for efficient resource use, the PCS/SERDES modules need to be in one Clarity Designer file and DDR modules in another

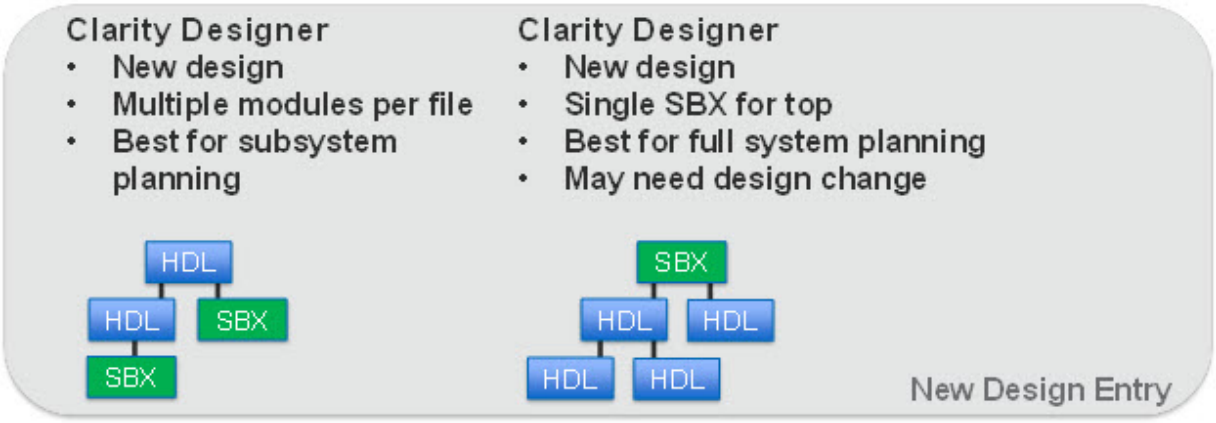
file. This means one module should be imported and then the existing Clarity Designer file should have the other modules copied and instantiated within it. See “Importing Existing PCS or DDR Modules or IP” on page 220 on steps needed for this.

Figure 26: Migrating Existing Designs in Clarity Designer



New designs can be optimized for efficient use in Clarity Designer. Modules that have a high amount of interconnection should be placed in the same Clarity Designer file. Modules at the same hierarchy level even though they are not interconnected can also be placed within the same file, as shown in Figure 27 on page 217. This will reduce the effort to configure, generate, and save them independently. The goal for new designs should be to use the tools most efficiently for your design structure rather than follow the same method needed for using IPexpress.

Figure 27: New Design Entry in Clarity Designer



Running Clarity Designer

To run Clarity Designer, first ensure that your device supports it. If it does not, it may support IPExpress instead.

To open the Clarity Designer startup dialog, do one of the following:

- Choose Tools > Clarity Designer

- Click  Clarity Designer in the toolbar

The tool can also be launched by choosing **File > New >**  **File** to create a new Clarity Designer file. Double-clicking an existing Clarity Designer file in the File List view will open that file and launch the tool. Only one Clarity Designer file can be open at a time.

Clarity Designer allows you to do the following tasks:

- Configure modules and IP from the Catalog tab
- Download and install new IP from the Catalog tab
- Import IPexpress (.ipx) files from an existing project and convert them into Clarity Designer format
- Import HDL modules from the project for use as modules within Clarity Designer
- Connect modules from the Builder tab
- Place PCS/SERDES and DDR elements into FPGA resources from the Planner tab

Using Single Module Generation The **Start Clarity Designer to generate a single Component SBX** option that appears when launching Clarity Designer provides a streamlined interface for generating single modules. This mode hides the Builder and Planner interfaces in Clarity Designer's main interface, as well as the DRC and Generate toolbar menus in Clarity Designer's toolbar menu. When customizing a module/IP, the Instance Path is editable and the Module Output does not have a default value.

To generate a module with Single Module Generation:

1. Open the Clarity Designer startup dialog
2. Select **Start Clarity Designer to generate a single Component SBX**
3. Click **Start**
4. Double-click an item in the Clarity Designer catalog to open the "Customize" dialog
5. Set the configuration options and click Configure to generate the module

   Clarity Designer will use this information to create and save the SBX file. If it does not exist in the Diamond File List, a pop-up dialog box will appear to confirm whether the file should be added to the File List.

To modify a module with Single Module Generation:

1. Open the Clarity Designer startup dialog
2. Select Open Clarity Design
3. Choose a Single Module/IP SBX file from the "Design File" list

4. Click Open

Note

You can also open a project by double-clicking it in the main window's File List tab.

5. Modify the configuration options
6. Click Configure

   Diamond will process the project file if the new SBX file is the project file. After the Configuration dialog closes, Clarity Designer will stay open in Single Component Mode.

Importing Existing Modules and IP from an Existing Design

Modules and IP that were generated with IPexpress and are in the project can be imported into Clarity Designer. They must be imported and configured, and the file generated and saved in the file list. Once the Clarity Designer (.sbx) file is saved in the file list, the existing IPexpress files must be deleted from the file list in order to successfully implement the design. The steps are shown below.

To import an IPexpress module:

1. Open Clarity Design using the toolbar, Tools menu, or by creating a new file.
2. In Clarity Designer, click the **Catalog** tab.
3. Click the **Import IP** tab (at the bottom of the view).
4. Click **Browse**.
5. In the Open IPX File dialog box, browse to the **.ipx** or **.lpc** file of the module. Use the .ipx if it is available.
6. Click **Open**.

   Most of the Import IP view is filled in.

7. Type in a name for Target Instance. This is the name that will be used for the configured module in the Clarity Designer project.
8. In the Target area, choose a Core Version. Usually you will leave it with the latest version.
9. Click **Import**.

   The module's dialog box opens.

10. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in the Lattice IP view for links to technical notes and user guides. Downloaded IP may come with additional information.

    As the options change, the schematic diagram of the module changes to show the ports and the device resources the module needs.

11. Click **Configure**.
12. Click the **Generate Log** tab to check for warnings and error messages.
13. Click **Close**.
14. Click **Generate** in the Clarity Designer toolbar and import the Clarity Designer file into the File List view.
15. Delete the original IPexpress file from the File List view.

The module or IP is now configured and is available in the Builder tab for connections. It is recommended that the Clarity Designer file be saved and the process repeated for each existing IPexpress file. This will allow the original design hierarchy to be reused without changes. However, if any of the modules or IP use PCS/SERDES or DDR resources, use the procedure below.

Importing Existing PCS or DDR Modules or IP

Modules and IP that were generated with IPexpress and are in the project can be imported into Clarity Designer. They must be imported and configured, the file generated and saved in the file list. Once the Clarity Designer (.sbx) file is saved in the file list, the existing IPexpress files must be deleted from the file list in order to successfully implement the design. The steps are shown below for when an existing module or IP uses PCS/SERDES or DDR resources. This procedure is different because PCS modules or DDR modules should be in the same Clarity Designer file to allow proper planning of the resources in the FPGA.

To import a PCS or DDR module:

1. Open Clarity Design using the toolbar, Tools menu, or by creating a new file.
2. In Clarity Designer, click the **Catalog** tab.
3. Click the **Import IP** tab (at the bottom of the view).
4. Click **Browse**.
5. In the Open IPX File dialog box, browse to the **.ipx** or **.lpc** file of the module. Use the .ipx if it is available.
6. Click **Open**.

   Most of the Import IP view is filled in.
7. Type in a name for Target Instance. This is the name that will be used for the configured module in the Clarity Designer project.
8. In the Target area, choose a Core Version. Usually you will leave it with the latest version.
9. Click **Import**.

   The module's dialog box opens.
10. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in the Lattice IP view for

links to technical notes and user guides. Downloaded IP may come with additional information.

As the options change, the schematic diagram of the module changes to show the ports and the device resources the module needs.

11. Click **Configure**.
12. Click the **Generate Log** tab to check for warnings and error messages.
13. Click **Close**.
14. Repeat steps 2 - 13 for each existing module using the same resources (all PCS/SERDES modules should be in one Clarity Designer file and all DDR modules in another Clarity Designer file).
15. Click **Generate** in the Clarity Designer toolbar and import the Clarity Designer file into the File List view.
16. Delete the original IPexpress file from the File List view.

The modules or IP are now configured and are available in the Builder tab for connections. They will also show in the Planner tab to allow those resources to be placed. Consolidating multiple modules within a single hierarchy level in a Clarity Designer file may require the original design hierarchy to be modified. This will vary for different designs on a case by case basis. Although this may cause design changes, there is a benefit to doing this. By putting the DDR modules, for example, in the same file these resources can now be planned together and feedback given on whether a valid placement is possible. Previously, constraints would need to be entered and if the solution was not valid, later implementation engine stages would generate errors. This new methodology is a faster and easier solution to placing resources within an FPGA and getting feedback if the solution is possible.

Current Restrictions

Clarity Designer contains several restrictions in this release. The list below lists the general usage restrictions. Restrictions within each module or IP are dependent upon the FPGA target and the use of that module or IP.

- Using Clarity Designer file as the top level of a design
  - All modules and IP must be directly instantiated within this file. Modules cannot be instantiated within an HDL file.
  - Any HDL in the design must be imported into the file as a module. HDL files can contain a hierarchy of other HDL modules, but no Clarity Designer modules or IP can be instantiated within the HDL.
  - Clarity Designer files cannot be used as modules within another Clarity Designer file.
- Builder tab connections
  - Cannot set HDL parameters of user modules.
  - Cannot tie ports with constants.
  - Cannot connect ports to logic equations.
- Implementation process

- No IPexpress (.ipx) files are allowed in the File List view.
- Clarity Designer (.sbx) files must be generated before they can be put into the File List view.
- Clarity Designer files must be in the File List view in order to correctly implement the design.

IP or Module Generation

IP or Module configuration is accomplished in the Catalog tab. Existing modules and installed IPs are listed in the Lattice IP sub-tab in the Catalog tab.

Downloading Additional IP

All available modules are installed with the Lattice Diamond software installation. Additional IP are downloaded from a Lattice IP site within the Clarity Designer tool. To add additional IP, the Lattice IP Server tab is used to download and install them.

To download Lattice IP:

1. If you are working on Linux and this is the first IP you have ever installed, go to your home directory and, with a text editor, create an empty file named **ipsetting_l.lst**. Enter no text: the file must be empty and must have this name. If you find you already have a file with this name, do not change it. This is a text file listing your IP and their path names.
2. In Clarity Designer, click the **Catalog** tab.
3. Click the **Lattice IP Server** sub-tab, located at the bottom of the tool.
4. Click **IP** in the left pane.

   The software connects to the Lattice Semiconductor Web site.
5. Expand the folder tree and select the IP you want to download.

   Information about the IP appears in the right pane including links for additional information.
6. Do one of the following:

   - To download and install the IP, right-click the IP and choose **Install** in the pop-up menu.
   - To just download the IP for later installation, right-click the IP and choose **Download** in the pop-up menu.
7. Follow the on-screen instructions.

Installing a Downloaded IP

After downloading an IP, you must install it to make it available in Clarity Designer.

Installing on Windows To install the IP:

1. If Clarity Designer is open, close it.
2. Go to the location where you saved the IP.
3. Double-click the executable file (.exe).

   The Lattice Semiconductor Setup program opens.
4. Follow the on-screen instructions.
5. Open Clarity Designer to see the installed IP. Click the **Catalog** tab and then the **Lattice IP** tab. Then expand the **IP** folder tree.

Installing on Linux To install an IP that you downloaded earlier on Linux, the process depends on whether the download is a tar file (.tar.gz) or a shell file (.sh).

To install the IP from a tar file:

1. If this is the first IP you have ever installed, go to your home directory and, with a text editor, create an empty file named **ipsetting_l.lst**. Enter no text: the file must be empty and must have this name. If you find you already have a file with this name, do not change it. This is a text file listing your IP and their path names.
2. Go to the location where you saved the IP.
3. Enter the following in a command line:

   tar -zxvf <*filename*>**.tar.gz**
4. In Clarity Designer, click the Import User Configurable IP  button.
5. In the dialog box, browse to the installation directory for the IP and select **nodeinfo.cfg**.
6. Click **Open**.
7. To see the installed IP, click the **Catalog** tab and then the **Lattice IP** tab. Then expand the **IP** folder tree.

To install the IP from a shell file:

1. Go to the location where you saved the IP.
2. Enter the following in a command line:

   sh <*filename*>**.sh** <*Diamond install path*>**/bin/lin** [<*destination path*>]

   The <*destination path*> is the full path name of where you want the IP to be installed. If not included, the IP will be installed in the current directory.

   The first time you install an IP, the installer creates a file, ipsetting_l.lst, in your home directory. This is a text file listing your IP and their path names.

3. To see the installed IP, click the **Catalog** tab and then the **Lattice IP** tab. Then expand the **IP** folder tree.

Configuring Modules or IP

To customize a module or IP with Clarity Designer, go to the Catalog tab. After the module or IP has been configured, it will be available in the Builder tab and, if it uses PCS/SERDES or DDR resources, the Planner tab.

To configure a module or IP:

1. In the Catalog tab, double-click the IP that you want to configure.

   A dialog box opens with some general information about the IP. To get more information about the IP, see the About tab to the right in Clarity Designer.

2. In the dialog box, specify the Instance Name. Instance Name is the base name for the module's files (that is, with no extension).
3. Choose from the Module Output menu. This is how the IP will be coded.
4. Click **Customize.**

   The IP's dialog box opens.

5. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in Clarity Designer for links to technical notes and user guides. Downloaded IP may come with additional information.

   As the options change, the schematic diagram of the module changes to show the ports and the device resources the IP needs.

6. Click **Configure**.
7. Click the **Generate Log** tab to check for warnings and error messages.
8. Click **Close**.

   The IP is added to the Builder tab and, if it uses PCS/SERDES or DDR resources, to the Planner tab.

Building

After configuring modules and IP, they need to be connected to each other and to the top of the Clarity Designer subsystem. The Builder tab provides two ways to approach this task:

- The Components view provides a spreadsheet listing all of the modules and their ports. The columns show how the ports are connected.
- The Schematic view provides a schematic drawing of the modules and their ports. You can drag the parts to see how they fit together.

Click the tab at the bottom of the Builder view to select the mode that you want to start with. You can change mode anytime you want.

Once building is complete, the subsystem can be planned or generated and saved if no planning is required. Once the subsystem has been generated, the design can be implemented with the design flow in the Lattice Diamond software.

Building with the Components View

The Components view lists all of the modules in the Clarity Designer project along with all of their ports. For each port there are three fields:

- Connection is the name of the connecting net that you will define in Builder.
- Type is how the port was connected:
  - “un-connected” indicates the signal has not been connected yet.
  - “fabric” indicates a user-defined connection.
  - “internal” indicates a connection made in the Planner tab.
  - “pad-pin” indicates a connection to the I/O of the FPGA.
- Export is checked when the signal is connected to the top of the subsystem.

To connect a port to the top of the Clarity Designer subsystem:

- Select the check box in the Export column.

  The default name of the connecting net appears in the Connection column for the selected port. This will also be the name of the port in the Clarity Designer subsystem. The default name is based on the module and port names of the port. You can change the name by double-clicking it and typing over it.

To connect module ports to each other:

1. If you want to change a connection, right-click one of the ports and choose **Disconnect**.
2. Select the ports that you want to connect together. You can select several input ports if they will all use the same source. There can be no more than one output port selected. There can be zero output ports selected if one of the input ports has its Export box selected.

   Output ports have yellow arrows that point left. Input ports have dark blue arrows that point right.
3. Right-click the name of one of the selected ports and choose **Connect**.

   The default name of the connecting net appears in the Connection column for each of the selected ports. The default net name is based on the module and port names of the output port. You can change the net name by double-clicking it in any row and typing over it.

Building with the Schematic View

The Schematic view provides a schematic of the Clarity Designer module including a block for each module that you've added, external ports, and wires connecting them all. The Schematic view also has two additional panes:

- The Design pane provides a nested list of all modules, their ports, and the ports of the Clarity Designer module.
- The Property pane provides information about the selected object (in the General tab) and a list of related wires (in the Connections tab).

To connect a port to the top of the Clarity Designer subsystem:

1. Select the port.
2. Right-click in the schematic and choose **Export**.

   The default name of the connecting net appears in the Connections tab of the Property pane. This will also be the name of the port in the Clarity Designer subsystem. The default name is based on the module and port names of the port.

To connect module ports to each other using right-click:

1. If you want to change a connection, select one of the ports. Then right-click in the schematic and choose **Disconnect**.
2. Select the ports that you want to connect together. You can select several input ports if they will all use the same source. There can be no more than one output port selected. There can be zero output ports selected if one of the input ports has its Export box selected.

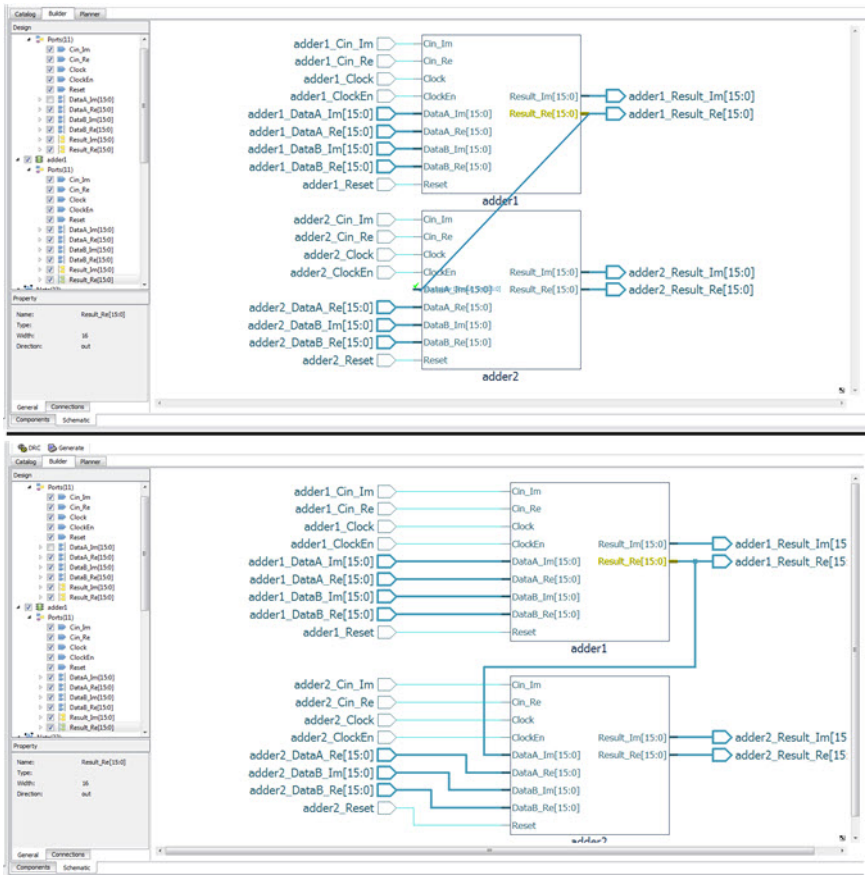
   In the Design pane, output ports have yellow arrows that point left. Input ports have dark blue arrows that point right.
3. Right-click the name of one of the selected ports and choose **Connect**.

   The default name of the connecting net appears in the Connection column for each of the selected ports. The default net name is based on the module and port names of the output port.

To connect module ports to each other using drag-and-drop:

1. If you want to change a connection, select one of the ports. Then right-click in the schematic and choose **Disconnect**.
2. Select the output pin of the port you wish to connect, and using the left mouse button, drag-and-drop it onto the pin of the input port you wish to which you wish to connect. A green check mark appears if the connection is allowed.
3. Release the left mouse button to make the connection. In the top portion of the example shown in Figure 28 on page 227, the adder1_Result_Im[15] output bus is dragged and dropped onto the available DataA-Im[15:0] input of adder2. The bottom portion of the figure shows the resulting connection when the left mouse button is released.

Figure 28: Drag and Drop Connections in Schematic Editor



PCS/SERDES Planning

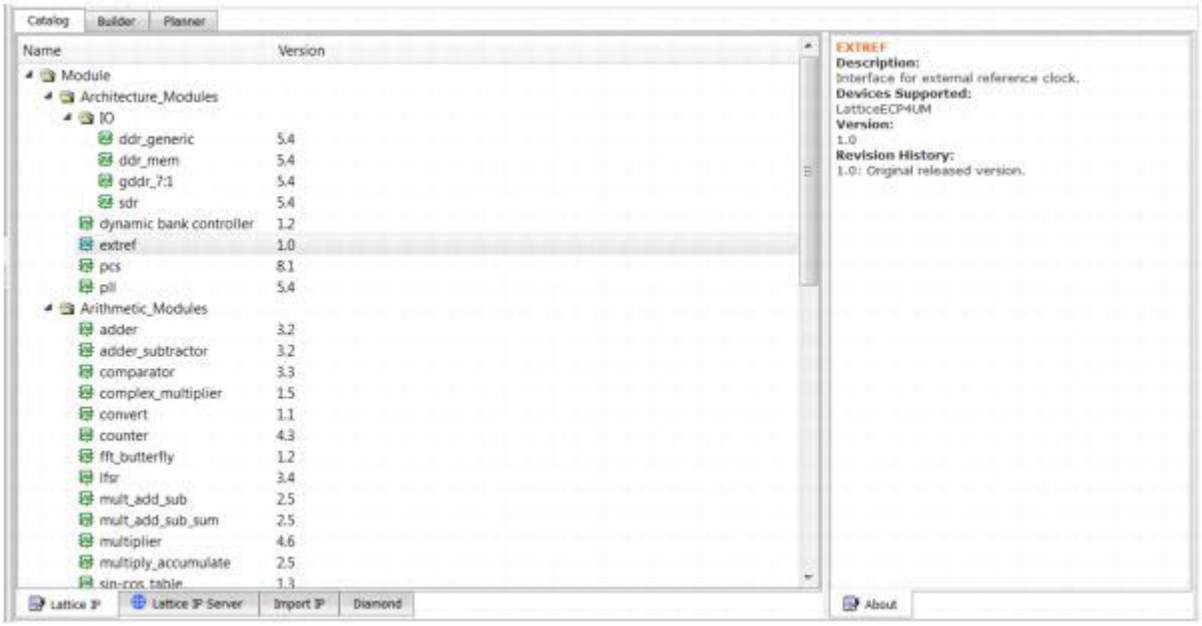
In order to plan the PCS/SERDES resources, the PCS or SERDES modules must be generated using the Catalog tab. If there is any connectivity between these modules, these connections must be completed in the Builder tab. “IP or Module Generation” on page 222 and “Building” on page 224 describe how these tasks can be accomplished. Once the modules are generated, the placement of the PCS or SERDES modules can be planned in the Planner tab.

PCS Configuration

An example for creating the SERDES Eye Demo using ECP5 is shown in Figure 29 on page 228. For this application both the Extref and PCS modules must be selected and configured from the Catalog tab.

Each module must be configured independently. Once both modules have been configured, they will appear in the Builder tab. They can then be connected to other modules in the subsystem or exported to the level above

Figure 29: PCS Configuration Example



for connection in RTL code to the rest of the design. See “IP or Module Generation” on page 222 and “Building” on page 224 for details on module configuration and building.

PCS Planning

To place the generated PCS module and the Extref modules, they can be dragged and dropped from the Resource listing on the left side of the Planner tab to the resources in the chip view on the right. The PCS-related resources are shown in the bottom section of the chip view as show in Figure 30 on page 229.

By placing the PCS module and the Extref, Clarity Designer will incorporate the placement of PCS module and the Extref block to the location automatically without any additional constraints or editor preferences.

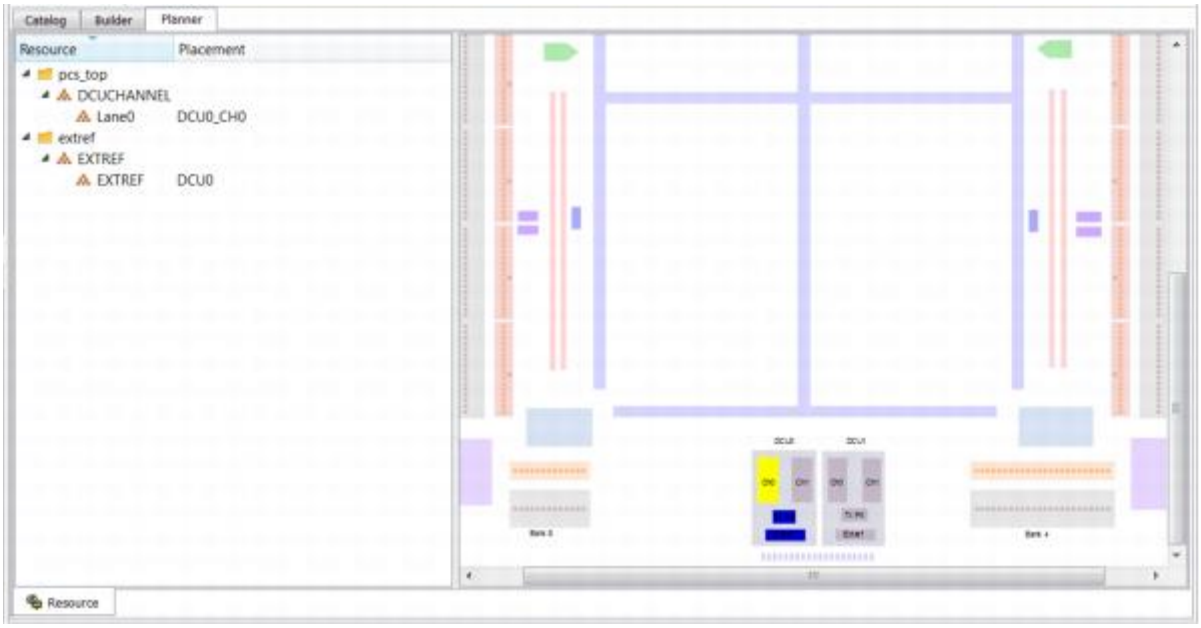
Once all PCS related resources are placed, then the DRC can be used to check that there are no DRC errors and use the generate button to complete the Clarify Designer subsystem and incorporate it into the Diamond project design flow.

Using Multiple PCS Modules in a Single DCU

To place multiple PCS channels in the same DCU, proper care must be taken to ensure that the two PCS blocks used in the design should be using the same external reference clock frequency and TXPLL settings for both the channels in the same DCU.

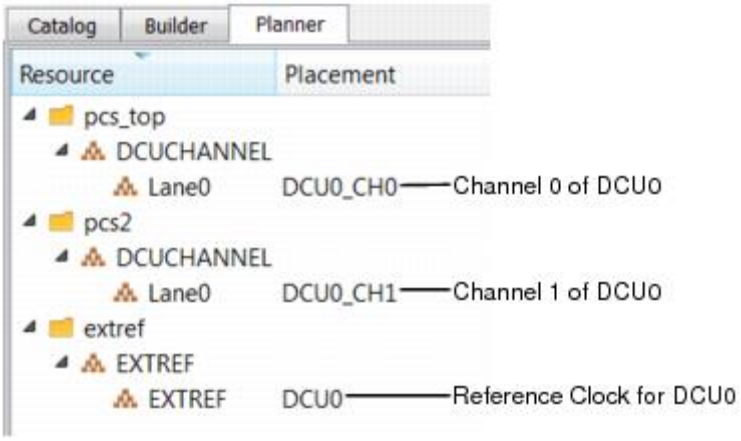
For example, a PCI Express x1 at 2.5 Gbps and a Gigabit Ethernet channel can share the same DCU using the half-rate option on the Gigabit Ethernet

Figure 30: PCS-Related Resources



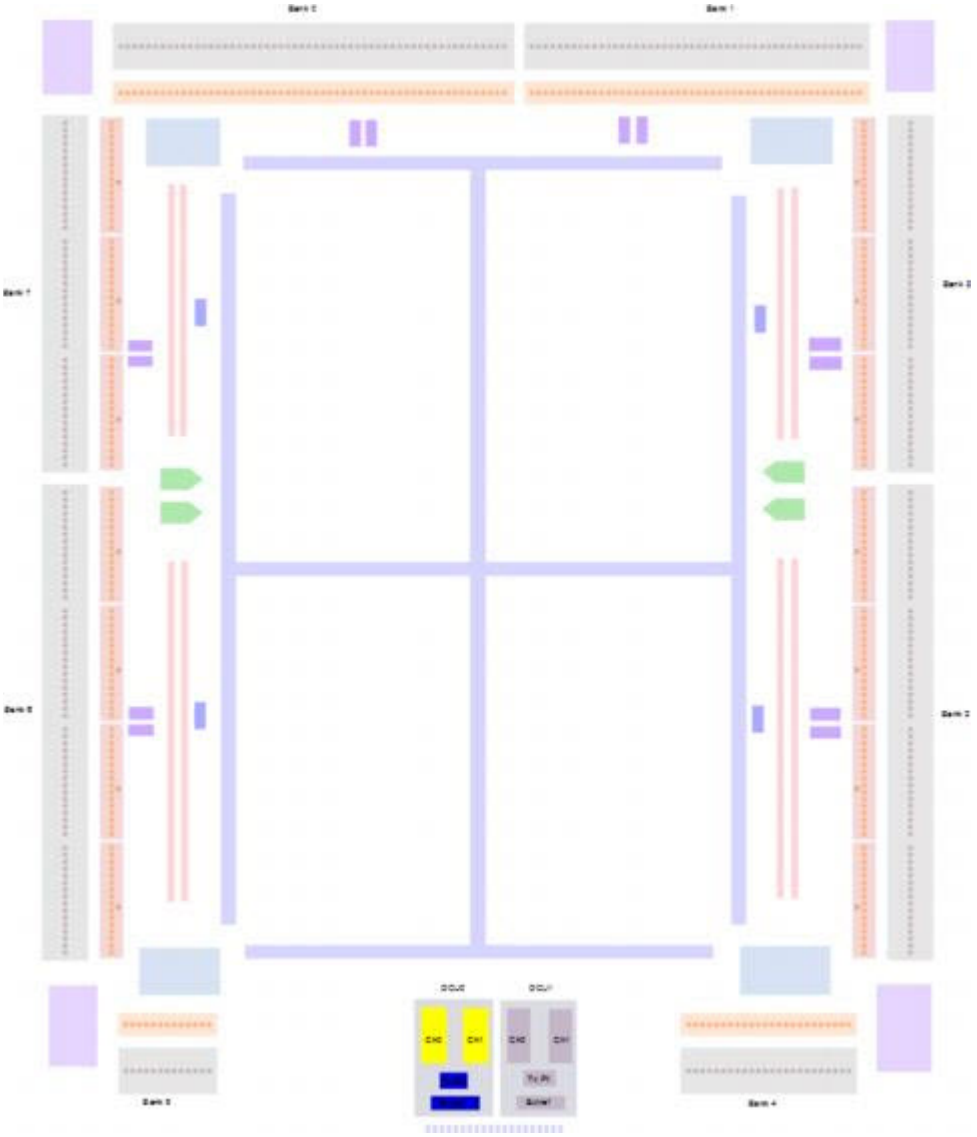
channel. When a DCU shares a PCI Express x1 channel with a non-PCI Express channel, the reference clock for the DCU must be compatible with all protocols within the DCU.

Figure 31: Using Multiple PCS Modules in a Single DCU



The placed DCU channels will be displayed as shown in Figure 32 on page 230.

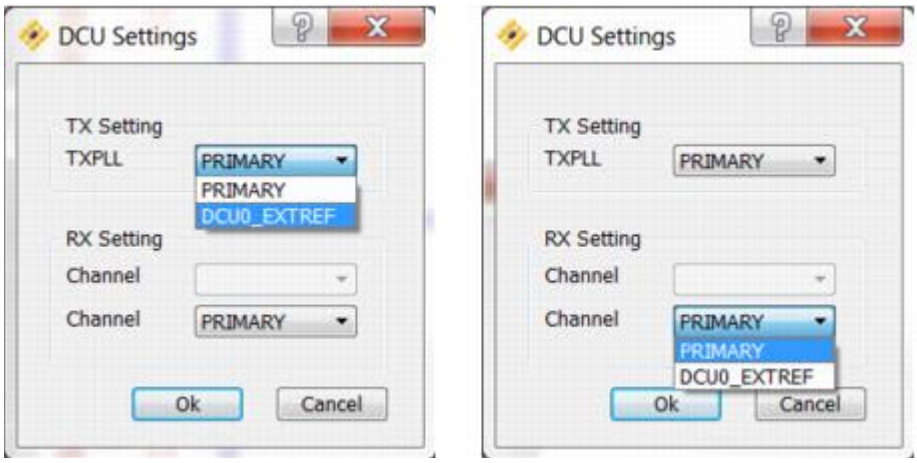
Figure 32: Placed DCU Channels



DCU Settings

In order to choose the TX and RX settings for the DCU, double-click the placed channel of the DCU and choose the input clock to the TXPLL inside and the reference clock for the RX side of the DCU as shown below:

Figure 33: Setting DCU Clocks



If there are multiple protocols that use the same reference clock or one protocol with a single reference clock frequency, choose the input to the TX and RX input clocks (TXPLL and RX setting clock) on the DCU. Either the Primary Clock input from the FPGA logic or a clock input from Extref components of DCU0 can be used.

Multiple Protocols on the Same Device

If multiple protocols have to be supported using a single ECP5 device that has different reference clocks, then the same DCU cannot be supported. A different DCU has to be implemented with its own external reference clock and Tx PLL settings with a PCS block to support multiple protocols as shown in Figure 34 on page 232.

The placed DCU channels will be displayed as shown in Figure 35 on page 232.

DCU Settings with Multiple Protocols

In order to choose the TX and RX settings for the DCU, double-click the placed channel of the DCU and choose the input clock to the TXPLL inside and the reference clock for the RX side of the DCU as shown in Figure 36 on page 233.

If there are multiple protocols, choose the input to the TX and RX input clocks (TXPLL and RX setting clock) on the second DCU. Use either the Primary Clock input from the FPGA logic or inputs from the Extref components of DCU0 or DCU1.

Figure 34: Using Multiple Protocols in the Same Device

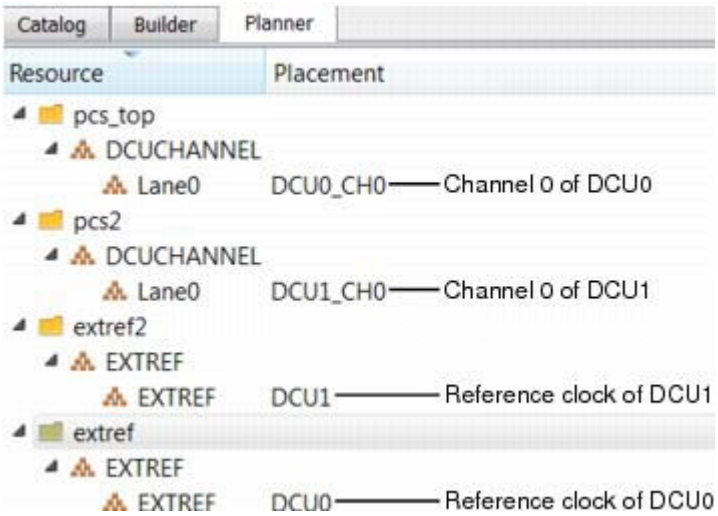


Figure 35: Placed DCU Channels

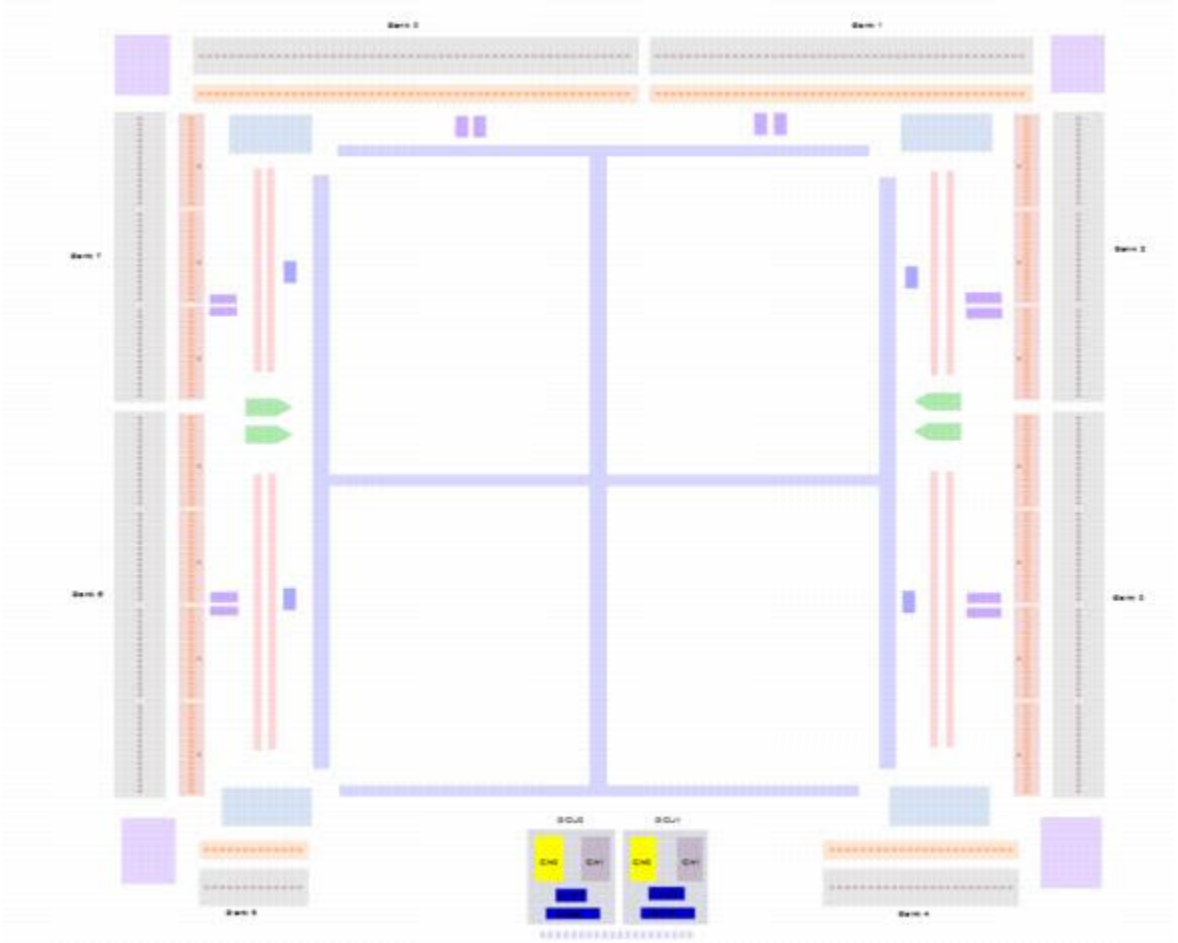
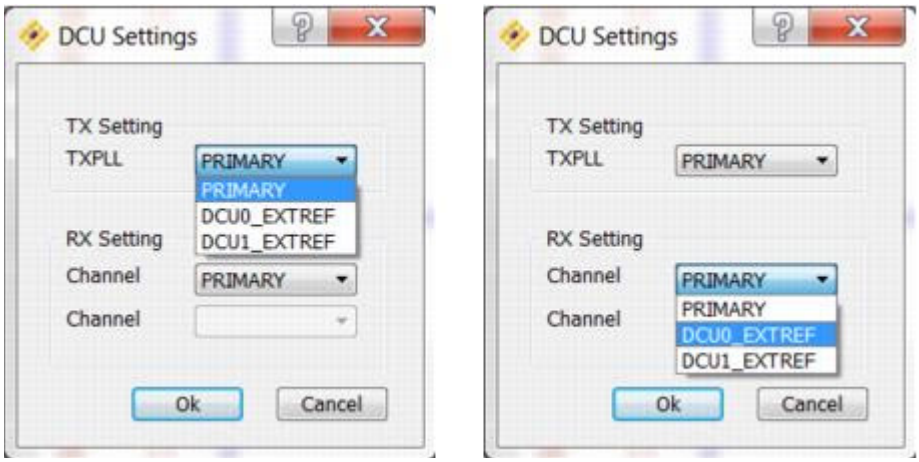


Figure 36: Setting DCU Clocks



DDR I/O Planning

In order to plan the DDR I/Os, the DDR modules must be generated using the Catalog tab. If there is any connectivity between these modules, these connections must be completed in the Builder tab. "IP or Module Generation" on page 222 and "Building" on page 224 describe how these tasks can be accomplished. Once the modules are generated, you can plan the placement of the DDR modules in the Planner tab.

Planner View for DDR Modules

Once DDR modules are generated, they are listed in the Planner. The resources required by each of the DDR modules are listed next to the module name. This includes the number of I/O required, number of PLLs used, and number of DLLs used by that module.

Each DDR module, when expanded, will show the clock groups under that module. The ports and primitives in each of the DDR modules is grouped based on the clocking requirement. All ports and elements clocked by the same clock are listed in a given clock group. This clock group could be a high speed "ECLK" (edge clock) group for interfaces that use a gearbox (x2 or x4) or a slower speed "SCLK" (primary clock) group for interfaces that do not use a gearbox (x1).

Each clock group will include:

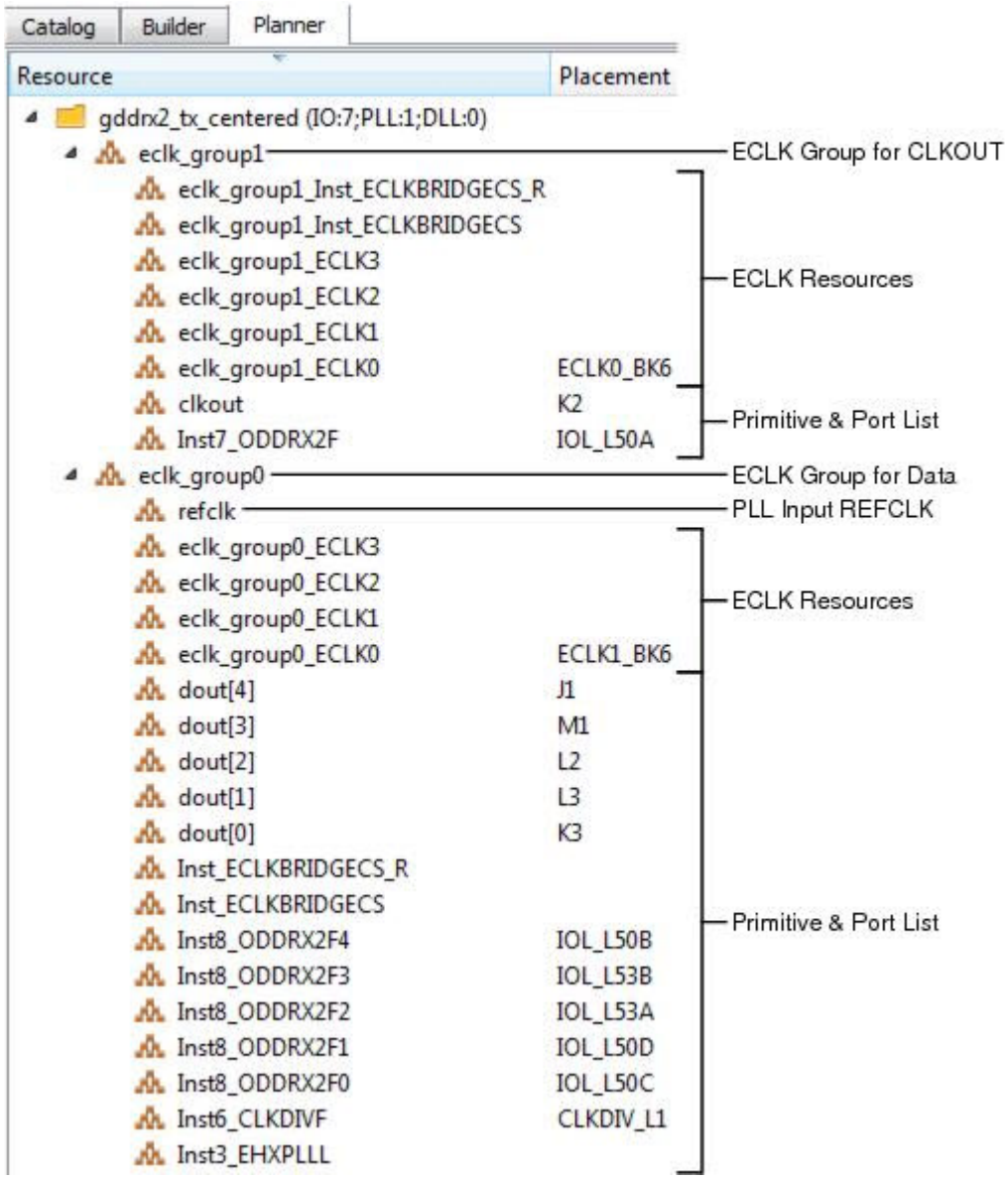
- Port list
- Primitive list
- ECLK and ECLKBRIDGE resources (if used by that module)

Once a port is placed, the Placement column shows pin numbers next to each of the ports and the ECLK resources shows the ECLK that was used for that interface.

The figure below shows how an interface will look when listed in the Planner tab. This interface is used for an example to show the clock grouping and port list. This Interface has two ECLK groups. Each group lists the ports clocked by that ECLK, the primitives that are clocked by that clock, and the ECLK resources.

Note that in this release, the ECLK resource section lists all the ECLKs and ECLKBRIDGEs available on the device even though only one ECLK is used in the clock group. Once the interface is assigned it will only show placement information for the ECLK that is used by that interface.

Figure 37: DDR Modules in the Planner View



Reconfiguration of an Interface

An interface can be reconfigured by right-clicking and selecting **Config**. This will allow you to change the interface parameters by opening the dialog box for that module.

Placing a DDR Interface

The following steps should be used to place any of the generated DDR interfaces.

1. Select the clock group under a given interface and then drag and drop onto the chip view. A valid location is indicated by a “+” symbol. Once the location is selected, the interface pins will be placed in a counterclockwise direction.

   The following also happens:

   - The pins used are displayed in the chip view.
   - The corresponding IOLOGIC cell will be highlighted.
   - If the group is an ECLK group, the ECLK tree used will be displayed.
   - The left pane will also display the locations of the Ports, ECLKs, and Primitives.

2. Drag and drop of a clock group will only place the data pins of that interface. The clock input pins in that interface will have to be placed manually. Expand the clock group and select the clock input pin. Drag it onto the dedicated clock pin. Right-click in the chip view to get the option to filter out all the dedicated clock pins.

   The following happens:

   - Once the clock pin is placed, the PLL, DLL, CLKDIV, and ECLKBRIDGE used by the DDR module will be displayed in the chip view.
   - The left pane will also display the locations of these modules.

If an interface cannot be placed in the location when the ports are dropped due to architectural limitations, all the data ports will be placed in the next available location. If it is a clock input port, Planner will not place the port and will issue a DRC message in the Output view.

DDR Interface Planning Guidelines

Here are some general rules to follow when placing DDR modules. Refer to the “High Speed Interface” technical note for the device for details on placement of each type of DDR interface using Clarity.

- On DDR interfaces, the data ports have to be placed before placing the clock input ports. Once the data ports are placed, manually drag and drop the CLKIN port to the desired location. The PLL, DLL, and CLKDIV used with the clock are not placed until the input CLK port is placed. This also applies to REFCLK input ports of PLL modules when generated with the DDR modules.

- On drag-and-drop of any interface, only data pins are placed. You have to manually place the clock input to any of the GDDRX interfaces as explained in “Placing a DDR Interface” on page 235.
- When placing the data ports, Planner does not use the dedicated clock sites on the device. These are reserved for the clock inputs.
- You must try to place all the data and clock ports of a given interface on one side. If there is a requirement to go around the corner to the other side, Planner uses the ECLKBRIDGE if it is available on that device. You must enable the ECLKBRIDGE when configuring the IP. Refer to the “High Speed I/O Interfaces” technical note for the device for details on how to use Planner to plan an interface with an ECLKBRIDGE.
- There are only a limited number of high speed clock (PLL, DLL, and ECLK) and I/O logic elements on each device. Planner does not allow interfaces to be placed if it runs out of resources.
- Planner always places the pins in a counterclockwise direction. You must drag and drop interfaces at the optimum location so that pins are not lost during planning. For example, if you require that all pins be placed in a single bank, the interface should be dropped at the first valid location of that bank counting counterclockwise so pins do not overflow to the next bank. Similarly, if all pins need to be on the same side, you must drag and drop at the first valid pin of the side. If you drag it into the edge of the side, the ports will automatically overflow to the other side, which forces Planner to use an ECLKBRIDGE when it is not required.
- When placing a GDDRX or DDR_MEM module that requires a PLL module, you can either choose to generate a PLL with the interface module or use a stand-alone module. If you choose to generate a PLL with the DDR interface, you must also choose the input buffer standard for the CLKI input of the PLL. This is required for Planner to correctly place the PLL. This will apply for GDDR.TX “centered” and DDR Memory modules.
- If planning the DDR Memory IP Cores, refer to the IP Core user manual for guidelines on placement of each IP module.

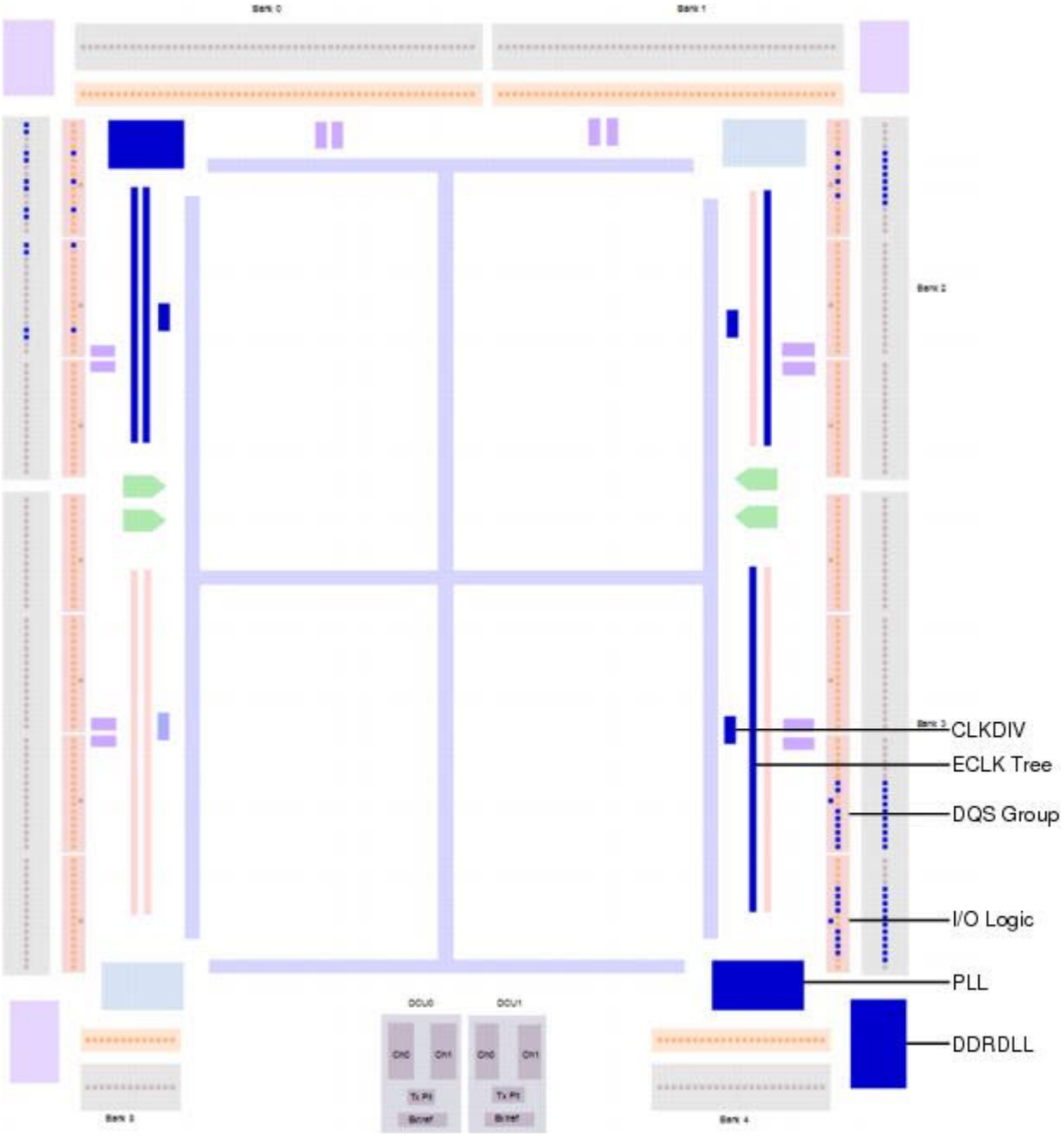
The chip view after placement of interfaces is shown in Figure 38 on page 237. This figure highlights some of the DDR resources that get used for an interface. For more details on the DDR resources, refer to the “High Speed I/O Interface” technical note for the device.

Viewing Interface Placement

Once an interface has been placed, you can view the placement by:

- Clicking on the port in the left pane to locate it in the chip view. The location of this port is highlighted in the chip view.
- Clicking on the placed pin in the chip view, which will highlight the port in the left pane.

Figure 38: DDR Resources in the Planner View



Resetting or Modifying Placement

Once the interface has been placed, it can be reset or modified using the following methods:

- If the entire interface placement has to be reset, right-click on the Interface name in the left pane and choose **Reset**.
- If the assignments for one of the clock groups under the interface have to be reset, right-click on the clock group in the left pane and choose **Reset**.

- If placement for a signal port has to be modified, choose the port in the left pane and drag and drop it to its new location.
- If placement for a signal port has to be reset, right-click on the port in the left pane and choose **Reset**.

Note

Clarity does not support multiple select on ports. The workaround is to modify placement of each port or modify placement of the entire clock group

Filtering Pin Types

You can get to the filter by either right-clicking on the chip view or on the toolbar. Planner allows filtering of dedicated PCLK pins, PLL/DLL clock pins, True LVDS pins, VREF pins, and so on. These will vary for each device. Refer to the “High Speed Interface” technical note for the device to see the options available.

Design Rule Check

Once all the interfaces are placed, you can click  **DRC** in the toolbar to see the list of warning and error messages.

Current Restrictions

- Planner does not see the signal connectivity in Builder. The workaround is to export the ports to the SBX file and make the connectivity outside of the SBX module. So any clock sharing between different DDR modules has to be external to the SBX.
- Stand-alone PLL are not available in Planner for placement or connection. Any connections to stand-alone PLL have to be done externally to the SBX.

Designing with Lattice Propel

Lattice Propel helps you produce system-on-chip (SoC) designs using a RISC-V processor. Propel has two parts: Propel Builder for the circuit design and Propel SDK for software development. Propel Builder provides the CPU, memory, and peripheral buses as customizable modules with an easy drag-and-drop interface. Propel SDK provides an industry-standard integrated development environment (IDE) to write and debug software.

Propel also comes with support for Platform Firmware Resiliency (PFR) in a LFMNX (Mach-NX) design.

After completing the SoC portion of the design with Propel, complete your design with Diamond.

Propel is a standalone tool that you need to download and install separately. To get Propel or learn more about it, go to www.latticesemi.com/latticepropel.

About LFMNX and Platform Firmware Resiliency

The LFMNX FPGA is designed to meet the enhanced security requirements of the National Institutes of Standards and Technology (NIST) SP 800-193 specification for Platform Firmware Resiliency (PFR). With LFMNX you can produce a “Root of Trust” (RoT) FPGA to insure that the firmware on a board does not become corrupted. For more information about the Lattice solution for NIST SP 800-193, see www.latticesemi.com/PFR.

Lattice Propel provides a PFR module to start your LFMNX design. The PFR module is a pre-configured and pre-verified hardened block.

Overview of the SoC Process

Start any System-on-Chip (SoC) design with Lattice Propel. When you’ve finished the SoC portion of the design, move the design into Diamond. Continue with the design, analysis, and testing as usual. See Figure 39.

To create an SoC design:

1. Install and open Lattice Propel.

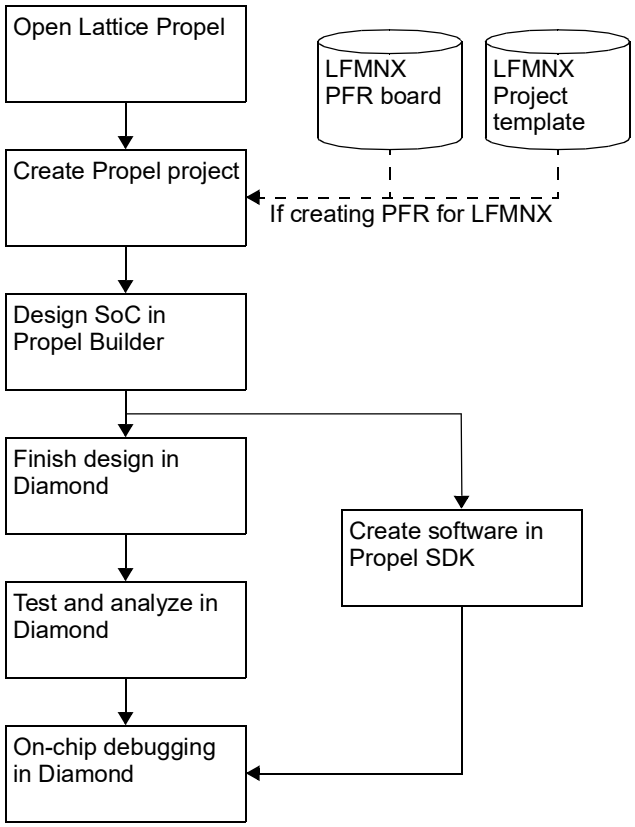
   Propel SDK opens.

2. Start a project.

   If you are creating a PFR design for LFMNX, specify use of the Raptor MCM board and the LFMNX Project template design. This design includes the PFR module.

3. Open Propel Builder to create the SoC.

Figure 39: Flowchart for Designing with Propel



4. When you have finished the SoC portion of your design, use Propel Builder or SDK to start Diamond.

   Propel opens Diamond and creates a project containing a collection of design files from Propel Builder. Propel Builder also creates a set of support files for Propel SDK.

5. Use Propel SDK to create software to run on the processor. Software development can be done in parallel with the Diamond portion of the FPGA design process.
6. Using Diamond, create and implement the rest of your design.
7. Analyze and test the design with Diamond.
8. If you are creating a PFR design for LFMNX, download code to the board's flash memory in the following order:
   a. PFR bitstream
   b. PFR firmware
   c. Rest of the design's bitstream
   d. User flash memory firmware

Designing with LatticeMico Platforms

Lattice Diamond allows you to import a platform that you have created in LatticeMico System. Such a platform includes a microprocessor and peripheral components that can be implemented in a Lattice FPGA device.

LatticeMico System is included as an optional installation with Diamond, through an open IP core licensing agreement. It can also be downloaded from the Lattice web site. LatticeMico System enables you to build the embedded processor platform using MicoSystem Builder and afterwards incorporate the platform's source files into a Diamond project. After generating the data file, you can use the Programmer tool to program the FPGA device or boot memory device on your target system.

For complete information about building a LatticeMico platform, see the Mico System Builder section of the LatticeMicoSystem online Help.

See Also

- "Adding a LatticeMico Platform" on page 39

About LatticeMico System

LatticeMico System is based on the Eclipse C/C++ Development Tools, which provide an open-source framework for building software. It contains two integrated tools and a debugger that combine with the Diamond software to build the microprocessor platform on a Lattice FPGA and write the software to drive it.

Lattice Mico System Builder Mico System Builder (MSB) enables you to generate a platform description and associated HDL for a hardware implementation, choose peripheral components to attach to the LatticeMico32 microprocessor or LatticeMico8 microcontroller, and specify connectivity between peripheral components.

After generating the platform in MSB, you can import the HDL files into Diamond, perform functional and timing simulation, place and route the design, and download the bitstream to the FPGA using the Programmer tool.

For complete information about using MSB to generate a platform, refer to the Mico System Builder section of the LatticeMico System online Help.

Lattice Software Project Environment and Debugger The Lattice Software Project Environment (SPE) enables you to develop the C or C++ application code that runs on platforms created with MSB, use the debugger to validate the code, and deploy the application code to on-chip or flash memory.

For complete information about using SPE to develop an application for the LatticeMico32 microprocessor or LatticeMico8 microcontroller, refer to the Lattice Software Project Environment section of the LatticeMico System online Help.

See Also

▶ “Adding a LatticeMico Platform” on page 39

Running LatticeMico System

If you are using Microsoft Windows, you can start LatticeMico System from the Windows Start menu. If you are using Linux, you can start LatticeMico System from the command line.

To open LatticeMico System using the Windows Start menu:

▶ Choose **Lattice Diamond > Accessories > LatticeMico System**.

To open LatticeMico System using Linux:

1. Go to the following directory:

   <*install_path*>/ispcpld/bin

2. Enter the command **`source setup_lv.csh`**.

   Or, if you are using the Korn shell, enter **`source setup_lv.sh`**.

3. Change to the LatticeMico installation directory.
4. Enter **LatticeMicoLauncher**.

LatticeMico System opens in a separate window. To create a new platform for your design, click the MSB button to open the Mico System Builder perspective.

See Also

▶ “Adding a LatticeMico Platform” on page 39

Schematic Design Entry

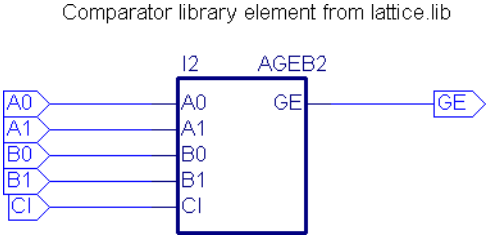
The Diamond schematic design environment is a set of tools that allows you to capture the structure of a design as either a flat description or a hierarchical set of components and the connectivity between these components. Then you can use this description to drive logic synthesis, implementation, and verification tools of the FPGA design flow. Schematics can be drawn on multiple “sheets” and be a variety of sizes.

Schematic Editor works in conjunction with Symbol Editor and Symbol Library Manager. You design by laying out symbols for basic functions and other modules and drawing wires between their ports. You can also create your own library of custom schematic symbols.

Schematic Overview

Unless you're using them just for documentation, schematics are actually the starting point of the development process, not the goal. The schematic will eventually be used to analyze the device's behavior using logic simulation and implement the logic into the target device using place and route. The Lattice Diamond software includes a Schematic Editor for creating and editing schematic sources. The figure below is an example schematic that represents 2-bit comparator gate.

Figure 40: Schematic Sample of a 2-bit Comparator



In the above schematic there are five inputs (labeled A0, A1, B0, B1, and CI) and one output (GE). The function of I2 is to perform a comparison of A[1:0] and B[1:0] with CI serving as carry-in.

In the FPGA design flow, Diamond creates a structural HDL model of the schematic in terms of component instances as in the following Verilog and VHDL.

Depending on the target device and version of the software, a symbol library may be provided or you may create your own symbols using Symbol Editor. A symbol library, lattice.lib, is provided for gate-level design of Lattice FPGA device families. The Lattice FPGA symbol library represents a sub-set of the library elements documented in the “FPGA Libraries Reference Guide” on page 1665. In addition, IPexpress automatically produces schematic symbols

Figure 43 on page 245 is a much more complex schematic.

Figure 41: Verilog HDL of a 2-Bit Comparator Schematic

```
/* Verilog model created from schematic sample.sch -- Feb 27,
2006 13:44 */
module sample( A0, A1, B0, B1, CI, GE );
 input A0;
 input A1;
 input B0;
 input B1;
 input CI;
output GE;
AGEB2 I2 ( .A0(A0), .A1(A1), .B0(B0), .B1(B1), .CI(CI), .GE(GE)
);
endmodule // sample
```

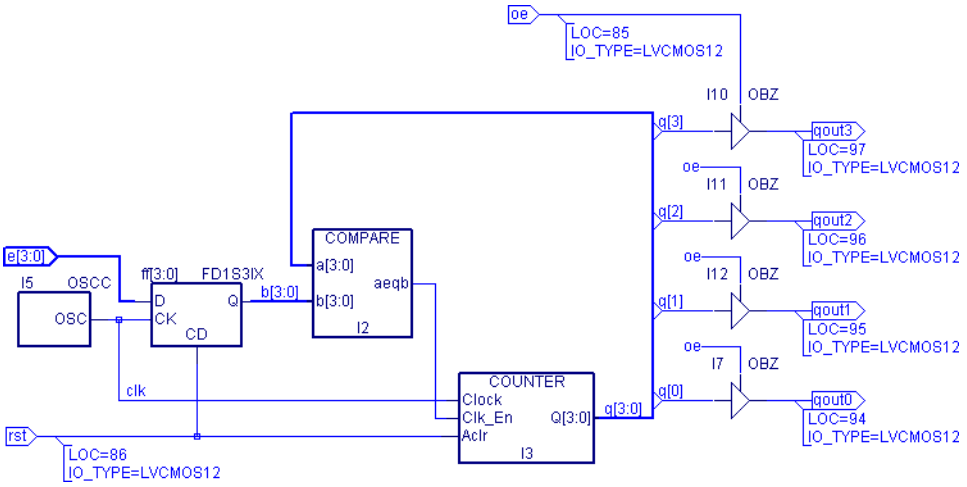
Figure 42: VHDL of a 2-Bit Comparator Schematic

```
-- VHDL model created from schematic sample.sch -- Feb 27
13:43:23 2006
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
entity SAMPLE is
      Port (      A0 : In    std_logic;
                  A1 : In    std_logic;
                  B0 : In    std_logic;
                  B1 : In    std_logic;
                  CI : In    std_logic;
                  GE : Out   std_logic );
end SAMPLE;
architecture SCHEMATIC of SAMPLE is
   SIGNAL gnd : std_logic := '0';
   SIGNAL vcc : std_logic := '1';
   component AGEB2
      Port (      A0 : In    std_logic;
                  A1 : In    std_logic;
                  B0 : In    std_logic;
                  B1 : In    std_logic;
                  CI : In    std_logic;
                  GE : Out   std_logic );
   end component;
begin
   I2 : AGEB2
      Port Map ( A0=>A0, A1=>A1, B0=>B0, B1=>B1, CI=>CI, GE=>GE
);
end SCHEMATIC;
```

What is a Schematic?

The schematic file (saved as a .sch file) describes your circuit in terms of the components used and how they connect to each other. The schematic can be used to create HDL structural models for logic simulation or synthesis. A schematic can represent a simple logic process (such as an AND gate) or a more complex component in your design (such as a memory function). It can also represent the top-level of your design.

Figure 43: Complex Schematic Sample



What do Schematics Consist of? A schematic is composed of the following items:

Symbols - These can be symbols from the standard symbol libraries, symbols representing other schematics you have drawn (Block symbols), or symbols you have created from scratch.

Wires - Wires connect the symbols. They can be single-signal (nets) or multiple-signal (buses).

I/O Ports - I/O ports show where signals enter or exit the schematic and the direction (polarity) of the signal. That is, whether it is an input, output, or bi-directional signal.

Attributes – Attributes are used to add functional characteristics such as user constraints to wires or symbols that are interpreted by tools like logic synthesis and place and route. Schematic attributes are translated into Verilog or VHDL attributes by the schematic netlister.

Graphics and Text - Graphics and text are usually added to display explanations. They are optional and have no electrical meaning.

A valid schematic must contain at least the first three components—symbols, wires, and I/O ports. For instance, a single, isolated component symbol cannot be the only element in a schematic. The schematic must include I/O ports for the external connections to the schematic, and these ports must be connected to the symbol with wires.

Symbols Symbols are graphic representations of components. The term “symbol” usually refers to an electrical symbol, such as a gate or a sub-circuit. You can draw graphic-only symbols (such as title blocks) with Symbol Editor, but these have no electrical meaning.

Symbols are the most basic elements of a schematic. Symbols represent primitive design elements, whether those elements are complete gates or a

Figure 44: A Valid Schematic

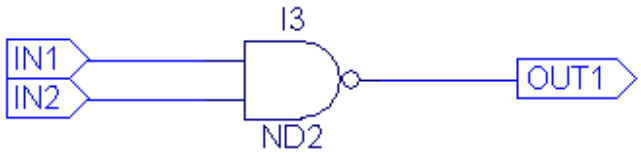
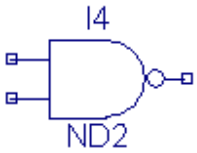


Figure 45: Invalid Schematic (No Wires or I/O Ports)



complex macro. A symbol can also be the hierarchical representation of a sub-circuit, or a “Block” symbol.

Each schematic symbol is a file ending with a .sym extension, and may be included in a library file with a .lib extension. The symbol file contains four types of information: graphics, text, pins, and attributes.

Pins Symbol pins are the connecting points between the symbol and the schematic wiring. If the symbol represents an individual component, the symbol pin represents the physical pin where a conductor can be attached. If the symbol represents a subcircuit (block symbol), the symbol pin represents a connection to an internal net of the subcircuit.

Wires Wires are the lines that electrically connect the symbol pins. Symbol pins are the only connection points for wires. You cannot connect wires to the symbol body itself.

There are two types of wires: single-wire nets and multiple-wire buses. Buses allow more than one signal to be routed as a single line.

Wires (both nets and buses) are added to schematics using the **Edit > Add > Wire** or the **Edit > Add > Net Name** commands of Schematic Editor.

There is only one kind of wire you can add (using the **Edit > Add > Wire** command). Whether it is a net or a bus depends on how you name the wire (using the **Edit > Add > Net Name** command). For instance, buses are named as <*busname*> [<*numberlist*>] where <*busname*> is the name of the bus and <*numberlist*> is an optional list of numbers separated by commas representing each net in the bus.

You would normally name all wires that connect to inputs or outputs and any internal nets with signals you want to view during simulation. You can use any name you like, but you usually choose a name that suggests the name or function of the signal carried by that wire. If you do not give a wire a name, Schematic Editor automatically supplies one, of the form N_<*n*> (where <*n*> is an integer).

Giving a single wire a compound name creates multi-wire buses. You can then tap off any signal you want anywhere along the bus.

Buses are most often used to group related signals, such as a 16-bit data path. However, a bus can be any combination of signals, related or not. Buses are especially useful when you need to route a large number of signals from one side of the schematic to the other.

Buses also make it possible for a single I/O port to connect more than one signal to a Block symbol. The signal names do not have to match, but both pins must carry the same number of signals.

I/O Ports I/O ports mark the points at which signals leave or enter the schematic. They are required. Any unconnected wire without an I/O port will eventually be flagged as an error when you try to create a netlist or run the simulator.

I/O ports are added to schematics using the **Edit > Add > IO Port** command in Schematic Editor.

The I/O port automatically takes the name of the wire it is attached to. If the wire is a bus, the port will have the same compound name as the bus.

When a Block symbol and its matching schematic are created, the I/O ports for the signals that enter and leave the schematic must have the same names as the corresponding pins on the Block symbol. The matching names identify which signal attaches to which pin.

Attributes Attributes of the schematic specify characteristics of the design that are interpreted by implementation tools in the design flow. Often attributes specify directives or preferences for logic synthesis or place and route in the FPGA design flow. See “Schematic Attributes” on page 283 for more details.

Each symbol has a number of predefined attributes that describe its component type and other unchanging characteristics. Other attributes can be given values after the symbol is placed in the schematic.

Like symbols and symbol pins, nets (the wiring that connects symbols to each other and makes external connections) can also have attributes. These attributes include the net's name, as assigned in the schematic.

Note

Net attributes for a sub-set of sysIO Buffer preferences are supported for the FPGA design flow.

Graphics and Text Graphics are pictures of the symbols. Symbol graphics have no electrical meaning and show only the position of the component in the circuit. A symbol's attributes and pins, not the graphics that represent it, define the electrical behavior of a symbol. Explanatory or descriptive text displayed with a symbol is also considered "graphic" information without electrical meaning.

Although symbols, wires, and I/O ports are visible, graphical items, they also have a functional or electrical meaning. In this context, "graphics" refers to the non-functional graphical parts of the schematic.

For example, you might add graphics showing the expected waveforms at different points in the circuit. Or, you could draw the company's logo and add it to each schematic for identification.

The most common use of graphics is to create a title block. The block shows the name and address of your company, and can include the company logo and blank spaces for the project name, schematic sheet number, and so on.

Title block symbols are available in the title.lib or lattice.lib libraries. You can modify the symbols in Symbol Editor and save the file (with the same name) in the project directory, or you can use Symbol Library Manager to add the revised symbol to your own custom symbol library file.

Text, like graphics, can provide additional information about the schematic or its project. Text can be placed anywhere on a schematic, even if it overlaps symbols or wires.

Text is added to schematics using the Edit > Add > Text command.

Naming Schematic and Symbol Files When you name schematic or symbol files, observe the following rules:

- Observe DOS file naming conventions. The dollar sign ($) cannot be the first letter of a file name, however.
- Do not type a file name extension; Schematic Editor will add it automatically. If you enter a non-standard extension, Schematic Editor will replace it with the correct extension (.sch for schematics, .sym for symbols, and .lib for symbol libraries).
- Symbol files must have the same name as the schematic file or HDL module that the symbol represents. For example, if you create a symbol for a Verilog module named "foo," the symbol's file must be named "foo.sym."
- Instances of a symbol in a design cannot use the same name as the symbol itself.

Symbols

Symbols are graphic representations of components. The term "symbol" usually refers to an electrical symbol, such as a gate or a sub-circuit. You can draw graphic-only symbols (such as title blocks) with Symbol Editor, but these have no electrical meaning.

Wires

Wires electrically connect schematic symbols. The symbol pins are the connection points for the wires. They can be single-signal (nets) or multiple-signal (buses).

I/O Ports

An I/O port is a special indicator that identifies a net name as a device input, output, or bidirectional signal. This establishes net polarity (direction of signal flow) and indicates that the net is externally accessible. An I/O port can either denote an I/O port of a design if it is in the top level schematic or denote an I/O marker in a lower level schematic.

Bus Taps

Signals enter or exit a bus at points called bus taps. A bus tap can be added to any existing bus, net, or wire. If a net or wire is not already a bus, adding the tap automatically promotes it to a bus.

Selecting the Type of HDL Files Generated for Schematic

To select the HDL file type generated from your schematic:

1. In the File List view, right-click the Project name and choose **Properties**.

   The Project Properties dialog box opens with the project name is selected at the top of the left pane.

2. In the right pane of the Project Properties dialog box, find Schematic HDL Type and select its Value cell.
3. Click the Value cell again to reveal a menu. Choose the desired HDL type. The default is Verilog.
4. Click **OK**.

Basic Schematic Editor Operation

Schematic Editor uses an *action–object* command structure. That is, you select the action you want to perform (usually from a menu), and then you select the object you want to act on.

For example, to move a symbol from a schematic, you first select the Move command from the Edit menu. Then you drag the symbol to move it.

Almost all commands remain in effect until you select a different command. For example, if the Add Wire command is active, you can continue to draw wires until you select a different command.

Edit Menu

The Edit menu consists of the following commands:

Table 5: Edit Menu Commands

Command	Description	Available	Shortcuts
Undo	Allows you to reverse the edit process. The Undo command operates on all commands that change a schematic or symbol (such as Cut, Paste, Add, Move, Drag, Attribute Window, Object Properties, and Clear), but it does not reverse those commands that do not change the drawing. The Undo command backs up one event each time it is issued. It can reverse all changes since the file was opened or saved. Saving a file discards the record of the preceding changes, and they cannot be undone or redone.	Schematic Editor Symbol Editor	Toolbar: Keyboard: Ctrl+Z
Redo	Reverses the effects of the Undo command. The Redo command can go forward, playing back the undone events until it reaches the end of the Undo log (the point when the first Undo in the sequence was issued). The Redo log is discarded whenever the current version of a symbol or schematic is saved. Once the log is discarded, the Redo command cannot recover undone commands.	Schematic Editor Symbol Editor	Toolbar: Keyboard: Ctrl+Y
Cut	Removes data (nets, graphics, symbols, or text) from the schematic and places it in the clipboard. The previous contents of the clipboard are erased. The items can then be pasted from the clipboard into another schematic.	Schematic Editor Symbol Editor	Toolbar: Keyboard: Ctrl+X
Copy	Copies items from the schematic to the clipboard. The items can then be pasted in another schematic.	Schematic Editor Symbol Editor	Toolbar: Keyboard: Ctrl+C
Paste	Places the contents of the clipboard into the schematic. When you select this command, the pasted contents are attached to the mouse pointer. Click the mouse pointer to place the contents to the right spot.	Schematic Editor Symbol Editor	Toolbar: Keyboard: Ctrl+V
Find	Finds wire and symbol instances on the schematic.	Schematic Editor	Toolbar: Keyboard: Ctrl+F
Clear	Deletes the selected items from the schematic drawing. Deleted items are not copied to the clipboard and the clipboard's contents are not affected.	Schematic Editor Symbol Editor	Keyboard: Delete
Select All	Selects all the items in the schematic.	Schematic Editor Symbol Editor	Keyboard: Ctrl+A
Add	Opens a menu for adding objects. See “Add Menu” on page 252.	Schematic Editor Symbol Editor	
Move	Enables the Move mode. Moves one or more elements (wire or symbol) to a different location within the schematic. You can move items in three ways. See “Moving Symbols” on page 282.	Schematic Editor	Toolbar:

Table 5: Edit Menu Commands (Continued)

Command	Description	Available	Shortcuts
Drag	Enables the drag mode. Moves portions of objects while the rest of the object remains in place. The effect is to stretch or reposition the item, altering its basic shape or direction.	Schematic Editor	Toolbar: 
Move, Drag	In Symbol Editor, Drag has the same function as Move. After selecting an item, you can move the cursor to the selected item, press the right mouse button, and move the item to the desired place when the cursor becomes a cross.	Symbol Editor	
Mirror	Works with items already attached to the cursor with the Add > Symbol command, Edit > Paste, or Edit > Move. The object is reflected through an imaginary vertical line each time the Mirror command is selected. The Mirror command can be combined with the Rotate command to produce eight standard orientations. See “Transforming Symbols” on page 282.	Schematic Editor	Keyboard: Ctrl+M
Rotate	Works with items already attached to the cursor with the Add > Symbol command, Edit > Paste, or Edit > Move. The object is rotated 90° counter-clockwise each time the Rotate command is selected. The Rotate command can be combined with the Mirror command to produce eight standard orientations. See “Transforming Symbols” on page 282.	Schematic Editor	Keyboard: Ctrl+R
Highlight	Selects the whole wire if a segment of the wire has been selected.	Schematic Editor	Toolbar:  Keyboard: Ctrl+L
Sheet	You can control your sheets in this dialog box.	Schematic Editor	
Symbol Origin	Assigns an origin or changes the current origin. See “Setting the Symbol Origin” on page 277.	Symbol Editor	
Expand page	Increases the work area for the symbol. See “Expand Page” on page 253.	Symbol Editor	
Attribute Window	Selects the attribute names for the symbol. See “Adding an Attribute Window to a Symbol” on page 280.	Symbol Editor	
Symbol Type	Selects the symbol type. See “Schematic Symbol Overview” on page 270.	Symbol Editor	
Object Properties	Assigns attribute values for symbols and wires. See “Assigning Attribute Values in a Schematic” on page 288.	Schematic Editor Symbol Editor	

Add Menu

The Add menu consists of the following commands:

Table 6: Add Menu Commands

Command	Description	Available	Shortcuts
Select	Enables the selection mode, allowing you to select items on the schematic.	Schematic Editor Symbol Editor	Toolbar:
Zoom Area	Enables you to zoom in on the schematic by pressing the left mouse button and drawing a box around the area of interest. See "Using the Zoom and Pan Tools" on page 454.	Schematic Editor Symbol Editor	Toolbar:
Pan	Enables you to see another part of the schematic by clicking and dragging. See "Using the Zoom and Pan Tools" on page 454.	Schematic Editor Symbol Editor	Toolbar:
Wire	Adds wires between symbol pins. See "Wiring a Schematic" on page 291.	Schematic Editor	Toolbar:
Net Name	Gives each net an unique reference name. Net names can contain only letters, numbers, apostrophes (single quote), and underscores. Net names are not case-sensitive. When you save your schematic, Schematic Editor automatically assigns names to unnamed nets. These names are in the form N_*nn*, where *nn* is an integer. You can override the Editor-assigned names by assigning your own names. See "Entering a Net Name" on page 295.	Schematic Editor	Toolbar:
Bus Tap	Adds a bus tap to a bus, net, or wire. See "Adding a Bus Tap" on page 300.	Schematic Editor	Toolbar:
Symbol	Opens the Add Symbol dialog box, which lets you select a symbol and place it on a schematic. Symbols come from symbol libraries or folders. Click on the schematic to place the symbol. See "Adding Symbols to the Schematic" on page 280.	Schematic Editor	Toolbar:
Instance Name	Names an instance symbol. Opens the Add Instance Name dialog box where you can enter an instance name. Then click on the instance symbol to name it. The instance name cannot be the same as a symbol name in the Lattice schematic library. Each symbol that is placed in a schematic is called a symbol "instance." Each instance of a symbol must have an unique reference name. Schematic Editor automatically assigns names to unnamed symbols each time the schematic is saved. These names are in the form I_*nn*, where *nn* is an integer.	Schematic Editor	Toolbar:
IO Port	Adds an I/O port to a net. Before this operation, the net should already have a name. After you select an I/O port in the Add IO Port dialog box, click on the net's name, then the I/O port is placed on the schematic. See "Specifying Signal Direction" on page 297.	Schematic Editor	Toolbar:
Big Bubble	Adds a big bubble to the layout. See "Drawing Negation Bubbles" on page 276.	Symbol Editor	Toolbar:
Bubble	Adds a bubble to the layout. See "Drawing Negation Bubbles" on page 276.	Symbol Editor	Toolbar:

Table 6: Add Menu Commands (Continued)

Command	Description	Available	Shortcuts
Pin	Assigns pins that correspond to I/O markers on the underlying schematic and connects the device represented by the symbol to the rest of the circuit. See “Adding Pins to a Symbol” on page 278.	Symbol Editor	Toolbar: 
Arc	Adds arcs to the drawing. These arcs are graphics only and have no electrical meaning in either the schematic or symbol. You can draw as many arcs as you like until you select another command. An arc is drawn from a starting to an ending point. These points can be defined by dragging the mouse from the starting point to the end point. Then release the button. You see an arc that starts and ends on the selected points and passes through the cursor. Move the cursor to change the shape of the arc. When the arc has the shape you want, click to add the arc to the drawing. If you decide not to draw the arc, right-click anywhere in the window to discard the selected points and restart the command. If you move the cursor outside of the schematic’s borders, the arc disappears until you move the cursor inside. See “Drawing Circles and Arcs” on page 276.	Schematic Editor Symbol Editor	Toolbar: 
Circle	Lets you draw circles, which are graphics and have no electrical significance. You can draw as many circles as you like until you select another command. See “Drawing Circles and Arcs” on page 276.	Schematic Editor Symbol Editor	Toolbar: 
Line	Lets you add individual line segments to the drawing. These line segments are graphics and have no electrical meaning in the schematic or symbol. You can draw as many lines as you like until you select another command. See “Drawing Lines” on page 275.	Schematic Editor Symbol Editor	Toolbar: 
Rectangle	Lets you add rectangles to the drawing. These rectangles are graphics only and have no electrical meaning in the schematic or symbol. You can draw as many rectangles as you like until you select another command. See “Drawing Rectangles” on page 276.	Schematic Editor Symbol Editor	Toolbar:
Text	Adds fixed text to the drawing. Text is used for adding notes, title blocks, and other information to the symbol or schematic. Text has no electrical meaning. The size and justification of the text is set by parameters in the dialog box. See “Drawing Text” on page 276.	Schematic Editor Symbol Editor	Toolbar: 

Expand Page

The initial work area is 40 x 40 primary grids. After a symbol has been created, Symbol Editor provides a work area that is approximately twice the height and width of the symbol.

When a symbol grows too large to fit, choose this command to enlarge the work area. Its initial use increases the size to 80 x 80 Primary grids. Subsequent use adds increments of 20 grids, up to a maximum of 400 x 400 grids.

The extra area is added at the right and bottom of the symbol. If more space is needed on the left or top, choose the Move command to shift the entire symbol.

Pay attention that the origin is not part of the symbol. If you move the symbol, the origin does not move with it. Be sure to reposition the Origin if you move the symbol.

Hierarchical Design with Schematics and HDL

A design with more than one level is called hierarchical. A single-level design is referred to as being flat. Converting a section of circuitry to a block makes a flat design hierarchical. This is commonly referred to as “hierarchical” design.

The Lattice Diamond software supports full hierarchical design by using schematics, HDL modules, or a mix of both. Hierarchical structuring permits a design to be broken into multiple levels, either to clarify its function or to permit the reuse of functional blocks. For instance, a large, complex design does not have to be created as a single module. By using a hierarchical design, each component or piece of a complex design can be created as a separate module.

A design is hierarchical when it is broken up into functional blocks, or modules. For example, you could create a top-level schematic describing an integrated circuit. In the schematic, you could place a Block symbol (a Block symbol represents a functional block) that provides a specific function of the chip. You can then elaborate the underlying logic for the Block symbol as a separate schematic or as a separate HDL module.

The module represented by the Block symbol is said to be at one level below the schematic in which the symbol appears. Or, the schematic is at one level above the Block's module. Regardless of how you refer to the levels, any design with more than one level is called a hierarchical design.

Advantages of Hierarchical Design

The most obvious advantage of hierarchical design is that it encourages modularity. A careful choice of the circuitry that composes your module will give you a Block symbol that can be reused.

Another advantage of hierarchical design is the way it lets you organize your design into useful levels of abstraction and detail. For example, you can begin a project by drawing a top-level schematic that consists of nothing but Block

symbols and their interconnections. This schematic shows how the project is organized but does not display the details of the modules (Block symbols).

You then draw the schematic for each Block symbol. These schematics can also contain Block symbols for which you have not yet drawn schematics. This process of decomposition can be repeated as often as required until all components of the design have been fully described as schematics.

Breaking the schematic into modules adds a level of abstraction that lets you focus on the functions (and their interaction) rather than on the device that implements them. At the same time, you are free to view or modify an individual module.

Although there are many ways of “breaking apart” a complex design, some may be better than others. In general:

- Each module should have a clearly defined purpose or function and a well-defined interface.
- Look for functions or component groupings that can be reused in other projects.
- The way in which a design is divided into modules should clarify the structure of the project, not obscure it.

Hierarchy vs. Sheets

Creating a hierarchical design is not the same as creating a schematic with multiple sheets. In a schematic, you can add as many sheets as desired to extend beyond the original sheet. However, regardless of how many sheets you add, all the components of the design are still at a single level; all sheets are still contained in the same module.

Approaches to Hierarchical Design

Hierarchical designs consist of one top-level module. This module can be of any format, such as VHDL, Verilog HDL, or schematic. Lower-level modules can be of any supported sources and are represented in the top-level module by functional blocks or other “place-holders.”

Following are some rules you need to follow when creating a hierarchical design in Diamond for a FPGA design.

- The top-level source can be of any format, such as VHDL, Verilog HDL, schematic, or EDIF netlist.
- For hierarchical Schematic/VHDL or Schematic/Verilog HDL designs:
  - If the upper-level source is a schematic file, the lower-level source can be either a HDL file or a schematic file.
  - If the upper-level source is a HDL file, the lower-level source can only be a HDL file.
- For EDIF designs:

- Hierarchical EDIF design is not allowed.

You can create the top-level module first, or create it after creating the lower-level modules. For example, in Schematic Editor you can create schematic project components in any order and then combine them into a complete design. You can draw a schematic first and create a Block symbol for it afterwards; or you can specify the Block first and create the schematic for it later.

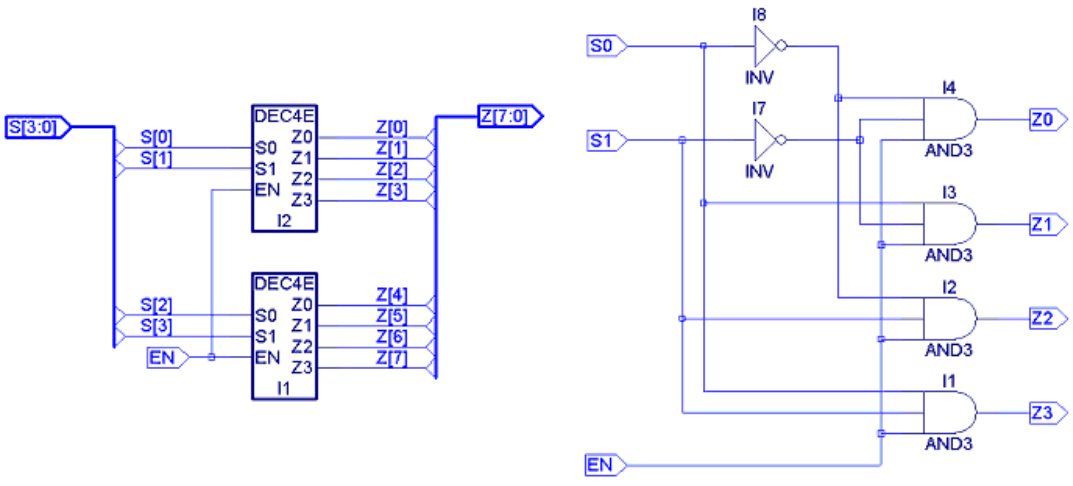
Hierarchical Design Structure Considerations

Apply the following considerations when using hierarchical design techniques. We take a hierarchical schematic design as an example (shown below). The rules implied in the example are also applicable to other types of hierarchical designs.

Hierarchical Design Structure The example below shows a 8-bit decoder circuit (DEC8E) constructed from two DEC4E symbols. The right side of the figure shows the underlying components. The two decoder symbols represent a total of eight AND gates and four inverters.

This hierarchical building process could be repeated by using Schematic Editor's **Design > Generate Symbol** command to create a symbol for schematic DEC8E, and then placing the DEC8E symbol in a higher-level schematic. If you created a schematic for a 16-bit decoder, DEC16E, by placing two copies of symbol DEC8E, you would be defining a circuit with a total of 24 gates. But instead of having to view 24 gates on a single level, you can work with symbols that represent gates, at the appropriate level of detail.

Hierarchical Naming In the DEC4E schematic example below, an AND3 gate has the instance name I1. In schematic DEC8E, two copies of the symbol DEC4E are placed and assigned instance names I1 and I2. Schematic DEC8E, therefore, contains two copies of AND3 gate I1.



Automatic Aliasing of Nets When a design is loaded, nets take the name of the highest (top-level) net in the design. That is, the name of top-level net propagates downward through the hierarchy to override the local name. By forcing all nets to the same name, this aliasing feature greatly speeds signal tracing in a multi-level design.

In the preceding example, the net name S[0] and S[1] of I1(DEC4E) is overridden by the higher-level external reference to become S[2] and S[3]. This override becomes the reference at all levels of the hierarchy.

Note

The hierarchical HDL output produced by the schematic system will retain the local names as defined by the schematic.

Creating a Hierarchical Schematic Design

You can generate a hierarchical design in the following way:

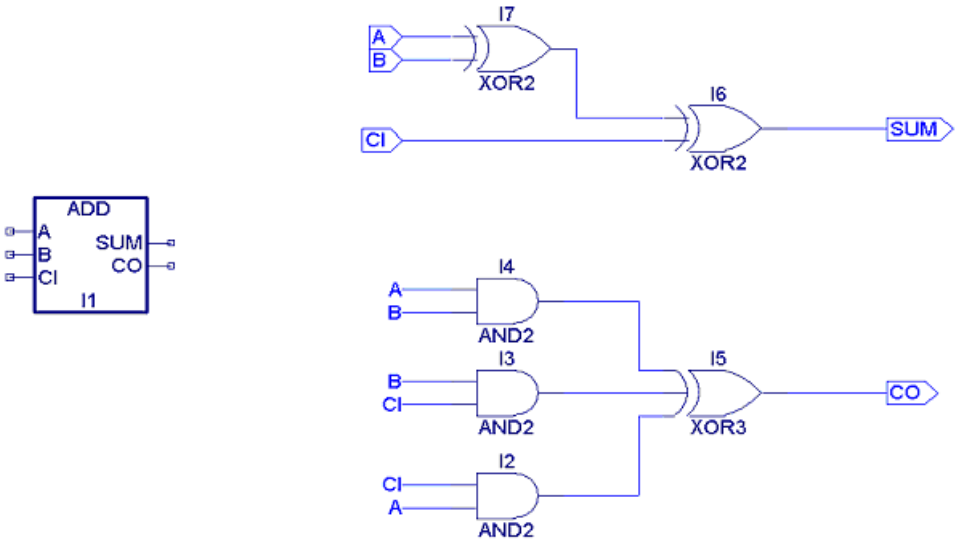
1. Create a schematic or HDL design for a lower level module.
2. Generate a schematic symbol for the lower level module. See:
   - “Creating a Block Symbol from a Schematic” on page 276
   - “Creating a Symbol from an HDL Module” on page 277
3. Store the new symbol in one of three ways:
   - Move the symbol to the same folder as the project file or the upper level schematic file.
   - Add the symbol to a library (see “Using Lattice Symbol Library” on page 260). Make sure the library is specified the User Defined Symbol Libraries strategy option. Choose **Tools > Options** to check. See “Adding Symbol Libraries to Schematic Editor” on page 269.
   - Add the folder of the symbol into the User Defined Symbol Libraries strategy option. Choose **Tools > Options** to check. See “Adding Symbol Libraries to Schematic Editor” on page 269.
4. Choose **Edit > Add > Symbol** to add the symbol to an upper level schematic. See “Adding Symbols to the Schematic” on page 280.

Schematic Hierarchy Example

The figure below shows an example of how a symbol corresponds to an underlying schematic. In this figure, pin A on the Block symbol corresponds to the net in the schematic, which is also named A. The other pins, B, CI (Carry In), CO (Carry Out) and SUM, also correspond to named nets in the schematic.

A Block Symbol and its Underlying Schematic

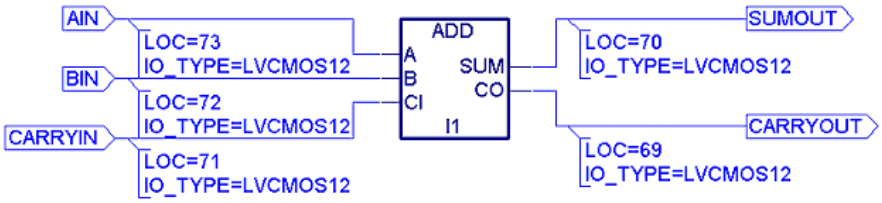
This following figure shows one top-level schematic with module I/Os and net attributes to specify sysIO Buffer preferences.



Top-level Schematic for Top (top.sch)

The name of the lower-level schematic must match the block name (schematic) in the upper-level schematic. This associates the lower-level schematic with the symbol representing it. The above schematic must be named add.sch.

The net names in the lower-level schematic corresponds to the pin names in the upper-level module. The symbol should be a Block symbol.



Beginning a New Schematic Design

To begin a new schematic design:

1. In the Diamond window, choose **File > New > File**.

   The New File dialog box opens.

2. In the New File dialog box, do the following:

a. Under Categories, select **Source Files**.

b. Under Source Files, select **Schematic Files**.

c. Specify Name and Location for the new schematic file.

d. Click **New**.

Schematic Editor opens with a new, untitled sheet.

Converting a Schematic Design to Another Device

You can first target this design to the new device. As some symbols are only valid for some specific devices, you may find that some symbols cannot be used for the new device. Replace these symbols with new symbols that support the new device, according to the list of Schematic Symbols. See "List of Schematic Symbols" on page 261.

Opening Schematic Editor

To open Schematic Editor, you can either open an existing Schematic Design (see "Opening an Existing Schematic Design" on page 259) or create a new schematic file (see "Creating a New Source File" on page 36).

Opening an Existing Schematic Design

You can open schematic design files through the File List view of a design project or directly, outside of any design project.

To open a schematic design from within a design project:

1. In the File List view, find the schematic file under the Input Files folder. The entry looks like this:

    *<path>*/*<filename>*.sch

2. Double-click the schematic file.

   Schematic Editor opens with the selected file.

To open a schematic design outside of any design project:

1. Choose **File > Open >**  **File**.
2. In the Open File dialog box, in the file type menu at bottom-right, choose **Schematic Files (*.sch)**.
3. Browse to the desired schematic file.
4. Click **Open**.

   Schematic Editor opens with the selected file.

Network Operation

When schematics, symbol files, and other project components are shared on a network, some form of overwrite protection is required. The system does not allow two people to work on the same file at the same time.

Schematic Editor uses the crash-recovery log file (described in the preceding section) to ensure single-user access. If you try to open a file and Schematic Editor finds a log file for it, you are asked if someone else is using the file. If you answer **Yes**, you cannot access the file.

If you answer **No**, you will be allowed to open the file, even if someone else is editing it. Check with anyone who might be using the file before answering **No**.

Using Lattice Symbol Library

The Lattice FPGA Symbol Library provides Boolean logic gates and synchronous elements for gate-level design. For macro-sized elements like architectural blocks, arithmetic, or memories it is recommended that you use the IPexpress interface to configure and generate schematic symbols and files for implementation. The Lattice schematic library provides dozens of library elements for gate-level schematic design.

The library file, lattice.lib, is composed of library elements compatible with most Lattice FPGA device families.

Symbol Library Manager allows you to manage libraries of symbols (.sym files) used in your designs. With Symbol Library Manager, you can view two types of symbol libraries: **folder** and **binary**. Folder libraries are simply directories that contain symbols. Binary libraries are .lib files that contain many symbols. You can clean up your folder structure and save disk space by organizing your symbols into binary libraries. Symbol Library Manager allows you to create your own binary libraries and maintain them by adding, extracting, deleting, copying, and renaming the symbols in the libraries.

Symbol Categories The Lattice FPGA Symbol Library for schematic design is based on a sub-set of the library elements documented in the “FPGA Libraries Reference Guide” on page 1665. The Libraries Guide is organized into functional categories by device family and provides details on function, I/O, and device compatibility.

For a complete list of all the symbols provided in the Lattice FPGA Symbol Library, see “List of Schematic Symbols” on page 261.

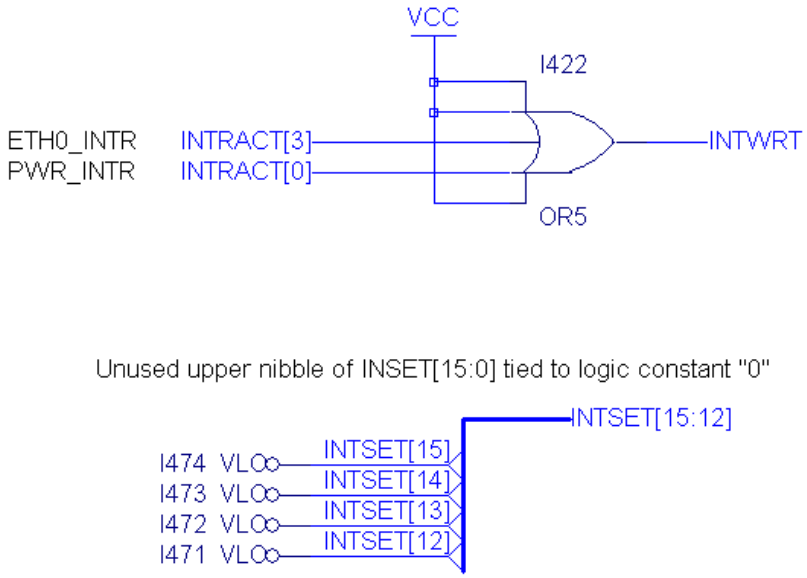
Attributes Schematic Editor for FPGA and MachXO Crossover Programmable devices supports the ability to specify HDL Attributes as schematic symbol attributes or net attributes. For more information, see “List of Schematic Attributes” on page 284.

Logic Constants Logic constants are a common construct in digital design to represent a static value.

Schematic Editor provides two special nets named VCC or GND that produce logic constants in the HDL output. To create a VCC or GND net, use the Edit > Add > Net Name feature to name a wire stub. Specify VCC or GND. Depending on the orientation of the wire, a special symbol for the net will appear as shown in the first sample circuit below.

The FPGA Symbol Library provides two symbols: VHI and VLO to represent bit value 1 or 0 respectively. To create a multi-bit value constant, use a combination of buses and symbols as shown in the second sample circuit below.

Figure 46:



Bus Declarations Schematic Editor allows you to create discontinuous bus ranges, for example MyBus [15:12] and MyBus [3:0], in separate regions of the schematic design. To avoid semantic problems in the resulting Verilog HDL or VHDL model of the translation process, the HDL model writer will create one bus declaration to cover the range: MyBus [15:0].

I/O Cell Usage The Lattice FPGA Symbol Library contains several I/O cell library elements (such as IB, OB, and BB) to help define the sysIO Buffer interface to the design. However, in general, it is not necessary to add these library elements to your schematic design. The logic synthesis process (Precision RTL or Synplify Pro) will automatically add them.

I/O Buffer Usage The Lattice FPGA Symbol Library for schematic design is based on a sub-set of the “FPGA Libraries Reference Guide” on page 1665.

List of Schematic Symbols

The Lattice schematic library, lattice.lib, is composed of library elements compatible with most Lattice FPGA device families. The following table lists all the schematic symbols included in the library file. Click the symbol name to learn its functionality and device compatibility.

In addition there are four more title blocks available in title.lib: FPTITLE, REV_BLK, SPTITLE, STITLE.

Table 7:

Category	Library Element	GSR Effect
Logic Gates	AND: AND2, AND3, AND4, AND5 NAND: ND2, ND3, ND4, ND5 OR: OR2, OR3, OR4, OR5 NOR: NR2, NR3, NR4, NR5 XOR: XOR2, XOR3, XOR4, XOR5, XOR11, XOR21 XNOR: XNOR2, XNOR3, XNOR4, XNOR5	
Miscellaneous	Carry-Chain: CCU2B Delay: DELAY Inverter: INV Logic High: VHI Logic Low: VLO Readback controller: RDBK Synchronous Release Global Set/Reset Interface: SGSR	
Comparators	A >= B: AGEB2 A <= B: ALEB2 A /= B: ANEB2	
Adders/Subtractors	Adder: FADD2, FADD2B Subtractor: FSUB2, FSUB2B Adder/Subtractor: FADSU2	
Multipliers	Multiplier: MULT2	
Counters	Up Counter: CU2 Bidirectional Counter: CB2 Down Counter: CD2 CD4P3BX CD4P3DX CD4P3IX CD4P3JX	

Table 7:

Category	Library Element	GSR Effect
Loadable Counters	Loadable Bidirectional Counters:	
	LB2P3AX	Clear
	LB2P3AY	Preset
	LB2P3BX	
	LB2P3DX	
	LB2P3IX	
	LB2P3JX	
	Loadable Down Counters:	
	LD2P3AX	Clear
	LD2P3AY	Preset
	LD2P3BX	
	LD2P3DX	
	LD2P3IX	
	LD2P3JX	
	Loadable Up Counters:	
	LU2P3AX	Clear
	LU2P3AY	Preset
	LU2P3BX	
	LU2P3DX	
	LU2P3IX	
	LU2P3JX	
Flip-Flops	D-Type Flip-Flops (Positive-level enable):	
	FD1P3AX	Clear
	FD1P3AY	Preset
	FD1P3BX	
	FD1P3DX	
	FD1P3IX	
	FD1P3JX	
	D-Type Flip-Flops (No enable input):	
	FD1S3AX	Clear
	FD1S3AY	Preset
	FD1S3BX	
	FD1S3DX	
	FD1S3IX	
	FD1S3JX	
	Loadable D-Type Flip-Flops:	
	FL1P3AY	Preset
	FL1P3AZ	Clear
	FL1P3BX	
	FL1P3DX	
	FL1P3IY	
	FL1P3JY	
	FL1S3AX	Clear
	FL1S3AY	Preset

Table 7:

Category	Library Element	GSR Effect
Latches	Data Latches: FD1S1A FD1S1AY FD1S1B FD1S1D FD1S1I FD1S1J Loadable Data Latches: FL1S1A FL1S1AY FL1S1B FL1S1D FL1S1I FL1S1J	 Clear Preset Clear Preset
Multiplexers	2:1 Mux: MUX21 4:1 Mux: MUX41 8:1 Mux: MUX81 16:1 Mux: MUX161 32:1 Mux: MUX321 PFU Mux: PFUMX LUT6 Mux: L6MUX21	
I/O Cells	Input Buffers: IB, IBPD, IBPU Output Buffers: OB, OBCO, OBZ, OBZPD, OBZPU, OBW Bidirectional Buffers: BB, BBPD, BBPU, BBW LVDS Input Buffer: ILVDS LVDS Output Buffer: OLVDS	
PIC Flip-Flops	Input PIC Flip-Flops: IFS1P3BX IFS1P3DX IFS1P3IX IFS1P3JX ILF2P3BX ILF2P3DX ILF2P3IX ILF2P3JX Output PIC Flip-Flops (System Clock): OFS1P3BX OFS1P3DX OFS1P3IX OFS1P3JX	
PIC Latches	Input PIC Latches (System Clock): IFS1S1B IFS1S1D IFS1S1I IFS1S1J	

Table 7:

Category	Library Element	GSR Effect
Special Cells	Combinatorial Elements: ORCALUT4, ORCALUT5, ORCALUT6, ORCALUT7, ORCALUT8 Dynamic Clock Selection: DCS Internal Oscillators: OSCA, OSCC, OSCD, OSCE, OSCH Startup: STRTUP Tri-State All: TSALL Global Set/Reset: GSR Power Up Set/Reset: PUR	
Title Blocks	title_a, title_b, title_c, title_d, title_e	

Creating a New Symbol Library

One of the first things you will probably want to do with Symbol Library Manager is to create a binary library for storing your own symbols or for storing symbols specific to a design.

To create a new symbol library:

1. In the Diamond window, choose **File > New > File**.

   The New File dialog box opens.
2. Under Categories, select **Other Files**.
3. Under Source Files, select **Symbol Library Files**.
4. Specify the file name without any extension. The .lib extension is added automatically.
5. Click **Browse**.
6. In the Location dialog box, browse to where you want to put the file.
7. Click **Select Folder**.
8. In the New File dialog box, click **New**.

   Symbol Library Manager opens, allowing you to create a symbol library. Now you can open other libraries and add symbols to your new library.

See Also ▶"Adding Symbols to a Library" on page 266

▶ "Copying Symbols to a Library" on page 267

Opening an Existing Symbol Library

You can open an existing symbol library file from the Diamond main window. There are additional methods for symbol library files and folders from the Symbol Library Manager window.

To open a symbol library file:

This method is only for symbol library (.lib) files, not library folders. This method can be used from the Diamond main window or from a detached Symbol Library Manager window.

1. Choose **File > Open > File**. (**File > Open File** if in a detached Symbol Library Manager window.)
2. In the Open File dialog box, in the file type menu at bottom-right, choose **Symbol Library Files (*.lib)**
3. Browse to the desired symbol library file.
4. Click **Open**.

   Symbol Library Manager opens with the selected symbol library.

To open a symbol library (file or folder) that is in your search path:

1. Select an open Symbol Library Manager view and choose **File > Open From Search Paths**.

   The Open Library From Search Paths dialog box opens.
2. In the dialog box, select the library you want to open.

   Selecting a folder that contains more than one .sym file allows you to access all of those .sym files. However, selecting a folder that contains a .lib file will not let you edit that .lib file unless the file name is specified in the search paths.
3. Click **OK**.

   The selected symbol library opens and the first symbol in the list is selected.

To open a folder library that is not in your search path:

1. Select an open Symbol Library Manager view and choose **File > Open Folder**.
2. In the dialog box, browse to the desired folder.
3. Click **Select Folder**.

   The selected symbol library opens and the first symbol in the list is selected.

Adding Symbols to a Library

To add symbols to a library in Symbol Library Manager:

1. Open the library to which you want to add a symbol.
2. Choose **Edit > Add Symbol(s)** or right-click any symbol in the library and choose **Add Symbol(s)** from the pop-up menu.

   The Add Symbols to Library dialog box opens.

3. Go to the location of the folder that has the symbol you want to add, and then select the symbol. You can select more than one symbol at a time by using the Shift and Ctrl keys on your keyboard.
4. Click **Open**.

   Symbol Library Manager copies the symbol into the current library. The original symbols still exist and may be deleted if so desired.

Extracting Symbols from a Library

You can copy a symbol from a binary library to a folder library with the Extract Symbol(s) command.

To extract symbols from a library:

1. Open the binary library (.lib) from which you want to extract a symbol and select the symbol. You can select more than one symbol at a time by using Shift and Ctrl keys on your keyboard.
2. Choose **Edit > Extract Symbol(s)** or right-click the symbol and choose **Extract Symbol(s)** from the pop-up menu to open the dialog box.
3. Type the path where you want to extract the file, or click **Folders** to browse for the path.
4. Click **OK**.

   Symbol Library Manager copies the selected symbol from the binary library to the folder library. You can use the File > Open Folder command to verify that the symbol was, in fact, copied.

Copying Symbols to a Library

You can copy a symbol to either a binary library or a folder library with the Copy Symbol(s) command.

To copy symbols to a library:

1. Open the library from which you want to copy a symbol and select the symbol. You can select more than one symbol at a time by using Shift and Ctrl keys on your keyboard.
2. Choose **Edit > Copy Symbol(s)** or right-click the symbol(s) and choose **Copy Symbol(s)** from the pop-up menu.

   The Copy Symbols dialog box opens.
3. Type the path to the folder library or binary library, or click **Folders** or **Libraries** to browse for the path.
4. Click **OK**.

   Symbol Library Manager copies the selected symbol from the current library to the new library.

Deleting Symbols from a Library

You can delete a symbol from a binary or folder library.

To delete a symbol from a library:

1. Open the library from which you want to delete a symbol and select the symbol. You can select more than one symbol at a time by using Shift and Ctrl keys on your keyboard.
2. Choose **Edit > Delete Symbol(s)**, press the Delete key on your keyboard, or right-click the symbol(s) and choose **Delete Symbol(s)** from the pop-up menu.

   Symbol Library Manager deletes the selected symbol from the current library.

Renaming Symbols in a Library

There are times when you want to change the name of a symbol.

To rename a symbol:

1. Open the library that contains the symbol you want to rename and select the symbol.
2. Choose **Edit > Rename Symbol(s)** or right-click the symbol and choose **Rename Symbol(s)** from the pop-up menu.

   The Rename Symbol dialog box opens.
3. Type the new name for the selected symbol.
4. Click **OK**.

   The selected symbol will now appear with the new name.

Note

Symbol names will be converted to module names in HDL. It is recommended that you follow the module name rules in HDL when naming symbols.

Saving a Symbol Library

The Save File command in Symbol Library Manager saves the changes to the existing library file. You can use the Save File As command to save the library with another file name.

Symbol library files have the extension .lib. Below shows the steps of saving a symbol library with the Save File As command.

To save a symbol library with the Save File As command:

1. In Symbol Library Manager, choose **File > Save** <*filename*> **As**.

   The Save As dialog box opens.

2. Go to the folder in which you want to save the library.
3. Type in a base name for the library.

   Symbol Library Manager adds the .lib extension automatically when you specify the base name.
4. Click **Save**.

Viewing Symbol Properties from Symbol Library Manager

You can view properties associated with any symbol in a library with the Symbol Properties dialog box. However, you cannot edit symbol properties in this dialog box.

To view symbol properties from Symbol Library Manager:

1. Open a library and select the symbol whose properties you want to view.
2. Choose **Edit > Properties** or right-click the symbol and choose **Properties** from the pop-up menu to open the Symbol Properties dialog box.

The dialog box has three tabs: General, Pins, and Attributes. The General tab has information on the name of the symbol and the location of the library it is in. The Attributes tab displays various attributes of the symbol. The Pins tab displays pin attributes. A pin's attributes may be viewed by selecting the pin whose attributes you wish to view.

Adding Symbol Libraries to Schematic Editor

After creating a symbol library, you need to tell Schematic Editor about it. Then the library and its collection of symbols will appear in Schematic Editor’s Add Symbol dialog box.

To add a symbol library to Schematic Editor:

1. In the Diamond window, choose **Tools > Options**.
2. In the left-hand pane of the Options dialog box, look under Schematic Editor and select **User Defined Symbol Libraries**.

   The right-hand pane shows a list of user-defined symbol libraries. The libraries that come with Diamond are not shown.
3. Click **Add Library** (for a .lib file) or **Add Folder**.

   The Add Symbol Library dialog box opens.
4. Browse to the library file (.lib) or folder that you want to add.
5. Click **Open**.

   The selected library is added to the list.

6. To change the order of the libraries in the Add Symbol dialog box, select a library and click **Up** or **Down**.
7. When you finish, click **Apply** or **OK**.

Creating Schematic Symbols

You can use Symbol Editor to construct schematic symbols. Besides the various lines, arcs, and boxes needed to create a symbol. You can add text to give information about the symbol and its relationship with the rest of the circuit.

You can also use Schematic Editor to create a block symbol from a schematic.

Schematic Symbol Overview

Symbol Types Four types of symbols can be created in Symbol Editor. Use the Symbol Type command on the Edit menu to select symbol types.

- Cell Symbol

  Cell symbols represent the primitive cells (transistors, resistors, diodes, and so on) used to design integrated circuits. The pins on cell symbols are named for identification in netlists.

- Block Symbol

  Block symbols are used to build a hierarchical design. Each block symbol represents a schematic or HDL module at the next-lower level of the hierarchy. Bus pins are permitted on block symbols.

- Graphic Symbol

  Graphic symbols add information that is not part of the circuitry. Graphic symbols are typically used for tables and notes. No pins are associated with graphic symbols, and they are never included in the hierarchy or netlists.

- Master Symbol

  Master symbols are used for title blocks, logos, revision blocks, and other standardized graphic symbols. You can add text to a master symbol to display the company name, address, project description, date, and so on.

Symbol Elements Schematics are constructed from symbols. A symbol can represent any logic component. Symbols are connected with wires (in Schematic Editor) to create a complete schematic whose behavior can be verified and simulated.

A symbol is a picture; it has no inherent electrical meaning. Its electrical characteristics are supplied by attributes that describe the symbol's behavior. (The behavior of a Block symbol is described by the schematic file associated with that Block symbol.).

The Lattice Diamond software supplies you with an extensive set of symbols. You can also use Symbol Editor to create Block symbols that represent a complete schematic, or part of one.

Symbols are composed of graphic, pin, and attribute elements.

- Graphic elements are the picture of the symbol. They have no electrical meaning; they show only the location of the component in the schematic.
- Pin elements on a symbol are points where a wire can be attached. The pins and wires are connected between symbols to circuit elements.

  Buses cannot be connected to pins unless the pin is a bus pin. Only ordered buses can be connected to bus pins.
- Attribute elements are properties of a symbol, pin, or net. Attributes can describe how the Fitter will optimize the symbol, or where it is placed in the device.

Pins Symbol pins are connection points for wires. Pins on Gate, Component, and Cell symbols represent the connection points on the device (pins or pads). Pins on a Block symbol represent connections from one level of the hierarchy to the level below. Because Graphic and Master symbols do not represent electrical components, you cannot attach pins to them.

Pins are the only symbol elements restricted to locations on the Primary grid, since wires must begin and end on Primary grid points.

Bus Pins A bus pin is used to connect a bus to a symbol. Naturally, a bus pin must have as many nets or signals as the bus that connects to the pin.

One way to create a bus pin is to give a pin a name of the form:

bus_name[*index1*–*index2*]

Where *bus_name* is the name of an internal bus, and *index1* and *index2* specify the range of signals you want to connect. For example, if you need to connect nine signals, *index1* could be 5 and *index2* could be 13.

Alternatively, a bus pin can be defined by giving it a *compound name*—a list of bus names separated with commas (,):

```
name1, clk, mux[0-3], toggle
```

Bus pins are allowed only on Block, Cell, and Component symbols. When a bus pin is created on a Component symbol, the numbers of the physical pins must be specified in the symbol definition.

These pin numbers are a list of pins assigned to the pin attributes BusPin_A through BusPin_H. When assigning bus pins, the normal PinNumber pin attribute *must not* have an assigned value.

The pin list can be divided sequentially among the eight attributes. Each individual attribute can hold about 200 characters. The list is delimited with

commas or spaces, and can specify sequences of pins in parentheses **( )** or square brackets **[ ]**. Examples are

```
BusPin_A = 1, 3, 5, (7:10)
  7 pins: 1, 3, 5, 7, 8, 9, 10
BusPin_B = A1 B[2:4] C1
  5 pins: A1, B2, B3, B4, C1
```

Attributes A symbol and each of its pins can have attributes. An attribute has a name and a value.

- Symbol Attributes

  Symbol Attributes are characteristics or properties associated with a symbol. Examples of symbol attributes are *PartNum*, *InstName*, *Width*, and *Type*.

  You can edit symbol attributes in the Object Properties Setting dialog box.

- Pin Attributes

  Pin Attributes are characteristics or properties associated with a pin. *PinName*, *Polarity*, and *PinNumber* are examples of Pin Attributes.

More on Pin Attributes Pin attributes generally have no meaning except as literal text. It is only when other programs read this text that it acquires any other meaning. There are a few exceptions:

- The Pin Name on Block symbols must match the name of the net connected to this pin on the underlying schematic in the hierarchy. The Pin Name can represent a single signal, or it can be a compound name representing a bus. All signals represented by a pin must exist on the underlying schematic.
- The Use or Polarity attribute has a default value of Input. You need to assign a value only if the Use or Polarity is other than Input. The other legal values are Output and Bidir (bidirectional).
- The values for Load and Drive attributes must be integers and use consistent units. Input pins use the attributes LoadLow and LoadHigh. Output pins use the attributes DriveLow and DriveHigh, while bidirectional pins use both the Drive and Load attributes.
- For gate symbols, pin numbers should be added in Symbol Editor as a list of pin numbers. For component symbols, a single pin number should be specified.

Opening Symbol Editor

To open Symbol Editor, you can either open an existing symbol (see “Opening an Existing Symbol” on page 273) or create a new symbol file (see “Creating a New Symbol File” on page 273).

Opening Symbol Library

To open a symbol library, you can either open an existing library (see "Opening an Existing Symbol Library" on page 265) or create a new library file (see "Creating a New Symbol Library" on page 265).

Opening an Existing Symbol

You can open an existing symbol file from Diamond's main window.

To open an existing symbol file:

1. In the Diamond window, choose **File > Open > File**.
2. In the Open File dialog box, in the file type menu at bottom-right, choose **Symbol Files (*.sym)**
3. Browse to the desired symbol file.
4. Click **Open**.

   Symbol Editor opens with the selected symbol.

Creating a New Symbol File

You can open Symbol Editor to create a new symbol.

To create a new symbol:

1. In the Diamond window, choose **File > New > File**.

   The New File dialog box opens.
2. Under Categories, select **Other Files**.
3. Under Source Files, select **Symbol Files**.
4. Specify the file name without any extension. The .sym extension is added automatically.
5. Click **Browse**.
6. In the Location dialog box, browse to where you want to put the file.
7. Click **Select Folder**.
8. In the New File dialog box, click **New**.

   Symbol Editor opens allowing you to create the symbol.

See Also ▶"Creating a Block Symbol from a Schematic" on page 276

▶ "Creating a Symbol from an HDL Module" on page 277

Saving a Symbol

You can store a symbol in a library for use in many designs, or keep it in the design directory for use in a specific design.

The Save File command in Symbol Editor saves the changes to the existing symbol file. You can use the Save File As command to save the symbol with another file name, which is useful when you're designing several similar symbols. Save the original, modify it, then save the new version with a new name.

Symbol files have the extension .sym. Below shows the steps of saving a symbol with the Save File As command.

To save a symbol with the Save File As command:

1. In Symbol Editor, choose **File > Save** <*filename*> **As**.

   The Save as Symbol File dialog box opens.
2. Go to the folder in which you want to save the symbol.
3. Type in a base name for the symbol.

   Symbol Editor adds the .sym extension automatically when you specify the base name.
4. Click **Save**.

Printing a Symbol

You can print a symbol using the Print command.

To print a symbol:

1. In Symbol Editor, choose **File > Print**.

   The Print dialog box opens.
2. Select the print options that you want and click **OK**.

Checking a Symbol for Errors

You can check symbols for errors at any time. Symbol Editor writes an error report to a file and opens it after the check is run. Clicking an error in the list highlights the error in the drawing.

The following types of errors are detected and reported:

- Block symbols should have a schematic with the same name in the current directory.
- Symbols of type other than Block are usually primitives and should not have a schematic of the same name in the current directory.
- Each pin should have a PinName in a Block or Cell, and a PinNumber in a Gate or Component.
- Pins in Component and Pin symbols can only have one PinNumber.
- Pins in the same group of a Gate must all have the same Polarity, Load and Drive.

- Pins on Block symbols should not have Load or Drive specifications.
- Pins on non-Block symbols should have Load or Drive specified. Input pins should specify Load but not Drive. Output pins should specify Drive, unless the pin is tristate. In that case, Load should also be specified, representing the load in the High-Z state. Bidirectional pins should specify both Load and Drive.

To check a symbol for errors:

1. In Symbol Editor, choose **Design > Drc Check**.

   Symbol Editor opens an error report.

2. Select an error from the list.

   The system displays the corresponding location of the error with a cross mark over it.

Drawing Operations

This section describes the operations that are generally applicable to drawing the graphic for a symbol.

Setting the Line Weight Graphic objects can be drawn in two line weights: normal and wide.

Normal lines (default) are the same width as the wires in a schematic. You might use this weight for drawing all elements.

Wide lines are twice the width of Normal lines, the same weight as schematic buses. For example, you might use this weight to illustrate a bus pin connection.

To set the line weight:

1. In the Diamond window, choose **Tools > Options**.

   The Options dialog box opens.

2. In the dialog box, click **Graphic Options** under Symbol Editor.
3. To draw wide lines, select **Use Wide Line**.

   All elements you enter are drawn as wide lines until you change this setting. Changing the line width does not alter the width of elements already drawn. It only affects the elements drawn after the change.

4. Click **Apply** or **OK**.

Drawing Lines When clicking to place the end points, lines are constrained to three principal directions: vertical, horizontal, and 45°. When you drag the line, the line can be at any angle as long as the end points fall on the grid being used. To show or hide the grid, choose **Tools > Options** in the Diamond window. Then select **Show Grid** in Symbol Editor Graphic Options.

Use the Add Line command to draw a line or an outline. When you draw a line, the line can be at any angle.

Drawing Rectangles Many symbols are based on a rectangular body. For non-rectangular symbols such as inverters or multiplexers, use the Line command to draw the outline.

Drawing Circles and Arcs You can place full circles with the Add Circle command and create portions of circles with the Add Arc command. Arcs are useful for the curved sections of NAND and NOR gates.

Drawing Negation Bubbles Negation bubbles are graphical and have no electrical significance. (Adding a negation bubble to a symbol does not change its logic. You must modify the symbol's attributes or the underlying schematic file.) You can add small or large bubbles:

- Click  on the Drawing Toolbar to draw a bubble one-half the Primary grid unit in diameter
- Click  on the Drawing Toolbar to draw a bubble one Primary grid unit in diameter

With either command, a bubble is attached to the cursor. Click the desired position in the schematic to place a bubble.

Drawing Text Text can be added anywhere in the drawing window. Typical uses of text include:

- Notes about the symbol
- Title blocks
- Cross references

Fixed text (as opposed to text appearing in attribute windows) can be drawn in up to eight sizes. (In the Windows version of the Diamond software, eight sizes are available. In the Linux version, three sizes are available.) Text can be left justified, right justified, or centered. Use the Add Text command to specify font size and text content and justification in the Add Text dialog box.

Creating a Block Symbol from a Schematic

Block symbols are used to build a hierarchical design. Each block symbol represents a schematic or HDL module at the next-lower level of the hierarchy. You can create a block symbol from the currently loaded schematic in Schematic Editor or for an HDL module selected in the Hierarchy view (see “Creating a Symbol from an HDL Module” on page 277).

To create a block symbol from the currently loaded schematic:

1. Open the schematic design in Schematic Editor.
2. Choose **Design > Generate Symbol**.

A generic block symbol is created at:
<*install dir*>/tmp/<*schematic name*>.sym.

3. Use Symbol Editor to customize the new block symbol.

See Also ▶"Adding Symbols to the Schematic" on page 280

Creating a Symbol from an HDL Module

The Lattice FPGA Symbol Library provides dozens of pre-defined symbols for gate-level schematic design. You can also create customer symbols (*.sym) from an existing Verilog or VHDL module. These symbols can later be added to your schematic design source.

Once a symbol is created, you can modify it in Symbol Editor.

To create a schematic symbol from an existing HDL module:

- In the Hierarchy view, right-click the module that you want to create a symbol for, and choose **Generate Schematic Symbol**.

  The Diamond software writes a new schematic symbol file (*.sym) into the project directory using the interface ports defined by your HDL module.

See Also ▶"Adding Symbols to the Schematic" on page 280

Preparing Symbols for Schematics

A symbol needs special *links* so that it can be recognized and placed in a schematic. These links consist of *pins*, *attributes*, *attribute windows*, and the *symbol origin*. You can define these links before adding symbols to schematics.

Setting the Symbol Origin

When a symbol is placed in Schematic Editor, the symbol is attached to the cursor. The point on the symbol attached to the cursor is called the *origin* of the symbol.

A newly created symbol has no origin. When the symbol is saved, the origin defaults to the upper-left corner of the symbol. You can assign an origin or change the current origin with the Symbol Origin command in Symbol Editor.

To set the symbol origin:

- Choose **Edit > Symbol Origin** and click the desired location. (The origin does not have to be on or within the symbol; it can be outside.)

  After you assign a new origin, the symbol coordinate tick marks relocate.

The Origin is not part of the symbol. If you move the symbol, the Origin does not move with it. Be sure to reposition the origin if you move the symbol.

Positioning Master Symbols

Master symbols cannot be freely placed on a schematic sheet. Instead, they are automatically positioned at one of the corners. This permits resizing the sheet without having to move the title block or other annotations.

The sheet corner is determined by the location of the symbol's origin. If the origin is placed at the upper-right corner of the Master symbol, for example, the symbol will be positioned at the upper-right corner of the sheet.

Master symbols do not have pins.

Adding Pins to a Symbol

Symbol pins correspond to I/O markers on the underlying schematic and connect the device represented by the symbol to the rest of the circuit.

Pins are usually attached to the symbol on short lines extending outward from the symbol's body. However, pins can be attached anywhere inside or outside the symbol, with or without connecting lines.

Pins are electrical elements and are therefore restricted to locations on major grid intersections. There can be only one pin at any location.

To add a pin to a symbol:

1. In Symbol Editor, choose **Edit > Add > Pin**, or click  on the Drawing Toolbar.
2. Click where you want to place a pin.

   If that point already has a pin, the system displays an error message. Pins appear in Symbol Editor as small squares.

Adding a Pin Name

You can use the Object Properties command to add pin names.

To add pin names:

1. In Symbol Editor, select the pin for which you want to add a name.

   The pin appears highlighted in red.
2. Choose **Edit > Object Properties**.

   The Object Properties Setting dialog box opens.
3. In the left-hand list, select the pin from the Pins list.

   The corresponding pin attributes are displayed in the right-hand table in alphabetical order.
4. Double-click the Value cell for the PinName attribute, and type in the name you want.

5. Click **Apply**.

Note

If you want the pin names to be displayed, use the Name Location field in the Object Properties Setting dialog box to set the display position of pin names. See "Setting the Display Position of Pin Names" on page 279 for details.

See Also ▶"Adding Pins to a Symbol" on page 278

Setting the Display Position of Pin Names

You can use the Object Properties Setting dialog box to control the pin name position relative to the pin.

To set the display position of pin names:

1. In Schematic Editor, select the pin you want to work with.
2. Choose **Edit > Object Properties**.

   The Object Properties Setting dialog box opens.
3. In the left-hand list, select the pin from the Pins list.
4. Under Name Location, select an option among the available choices: showing the pin name on the left, top, right, or bottom of the pin; or showing the pin name vertically. Also, you can set the offset value, the distance of the pin name from the pin.
5. Click **Apply**.

   The pin name appears at the specified position.

   Note

   The Offset is measured in Secondary grid units. The maximum Offset is 127.

See Also ▶"Adding a Pin Name" on page 278

Adding or Changing a Pin Attribute Value

Pin Attributes are characteristics or properties associated with a pin. *PinName*, *Polarity*, and *PinNumber* are examples of Pin Attributes.

To add or change pin attribute values:

1. In Symbol Editor, select the pin for which you want to add or change attributes.
2. Choose **Edit > Object Properties**.

   The Object Properties Setting dialog box opens.
3. In the left-hand list, select the pin from the Pins list.

The attributes for the selected pin are displayed in the right-hand table in alphabetical order.

4. Double-click the Value cell for the desired attribute, type in a new value, and click **Apply**.

   The attribute value is assigned to the selected pin.

See Also ▶"Pin Attributes" on page 272

Adding an Attribute Window to a Symbol

Attribute windows are predefined areas on or near a symbol or pin in which attribute values are displayed. In Symbol Editor, the window only contains the attribute name. When the symbol is instantiated in a schematic drawing, the attribute window contains the attribute value for that instance, rather than the attribute name.

Attribute windows do not have a visible outline. If no value is displayed, there is no indication that an attribute window has been defined.

To add an attribute window to a symbol in Symbol Editor:

1. Choose **Edit > Attribute Window**.

   The Attribute Window dialog box opens. It contains all attribute names.

2. Select the desired attribute name from the list.

   The attribute name is attached to the cursor.

3. Click to place the symbol attribute window at the desired position on (or near) the symbol.

   Two narrow bars appear beside the window's name, in a position indicating the text justification. For example, a right-justified name has the bars at the right end of the name.

4. Close the dialog box when you finish.

Adding Symbols to the Schematic

You can use the Add Symbol command on the Edit menu to select and add symbols to a schematic. Symbols come from a symbol library, or you can create your own symbols using Symbol Editor. Symbol files have the file extension .sym. Symbol libraries have the file extension .lib.

To select and add a symbol to a schematic:

1. In Schematic Editor, choose **Edit > Add > Symbol**.

   The Add Symbol dialog box opens.

2. In the dialog box, choose a library from the Library list.

   Tip

   If you want your own library of custom schematic symbols to appear in the Library list, you need to specify them in advance. See "Adding Symbol Libraries to Schematic Editor" on page 269.

3. From the Symbols list, select a symbol.

   The symbol is attached to the cursor.

4. After the symbol is attached to the cursor, click the location in the schematic where you want to place the symbol. If you pick the wrong symbol or change your mind, right-click anywhere in the window to remove the symbol from the cursor. You can then select a different symbol.

   Note

   The symbol remains attached to the cursor, so that you can place multiple copies without having to reselect the symbol.

See Also ▶"Using Lattice Symbol Library" on page 260

Deleting Symbols

To delete a symbol from a schematic:

1. In Schematic Editor, click the symbol that you want to delete.
2. Choose **Edit > Clear**.

   The symbol is deleted from the schematic.

Copying Symbols

If an instance of the symbol you want already appears in the schematic, you can copy it to another location.

To copy a symbol:

1. In Schematic Editor, choose **Add > Symbol**.
2. Click the instance that you want to copy.

   A copy of that symbol is attached to the cursor.

3. Click the location in the schematic where you want to place the symbol.

   Schematic Editor places a copy of the symbol at that location. The symbol remains attached to the cursor, so that you can place multiple copies without having to reselect the symbol.

Selecting Symbols

You can select one or more symbols in Schematic Editor. The selected symbols are highlighted in red.

To select symbols:

1. Choose **Edit > Add > Select**, or click on the drawing toolbar.
2. Do either of the following:
   - Click the symbol.
   - Draw a rectangle area around the desired symbols.

   You can continue to select symbols, in either of the above mode, as long as you hold down the Shift key.

Note

You cannot deselect a symbol by clicking it again. To deselect symbols, click any empty space on the schematic sheet.

Transforming Symbols

You can *mirror* or *rotate* symbols in Schematic Editor before you place them.

To mirror a symbol before placing it:

- When the symbol is attached to the cursor, choose **Edit > Mirror**.

To rotate a symbol before placing it:

- When the symbol is attached to the cursor, choose **Edit > Rotate**.

Moving Symbols

You can move symbols in Schematic Editor in three ways.

To move a single symbol:

1. Select from the drawing toolbar.
2. Choose **Edit > Move**.
3. Click the symbol that you want to move.
4. Drag it to a new location to move the symbol there.

Net names, attributes, and instance names are moved as well as the physical representation of the symbol. A priority list determines which item is selected if the cursor points at two or more items when the mouse button is clicked.

To move all symbols in an area:

1. Select from the drawing toolbar.

2. Choose **Edit > Move**.
3. Drag the mouse to define a rectangle area around the symbols that you want to move.

   When the mouse button is released, all items totally enclosed by the rectangle are selected.
4. Drag the selected symbols to a new location.

   All items totally enclosed by the rectangle are moved.

To move a group of symbols:

1. Select from the drawing toolbar.
2. Choose **Edit > Move**.
3. Select one or more items, as either a single item or area.

   As each item or area is selected, it is highlighted. You can continue to select items, in either single item or area-selecting mode, as long as you hold down the Shift key.
4. Drag the selected group to a new location.

   All items in the group are moved. The relative position of the items in the group is maintained.

Schematic Attributes

You use attributes to describe the characteristics or properties belonging to, or associated with, a *symbol*, *pin*, or *net*. Attributes only apply to describing characteristics in schematics. HDL source files have their own syntax for describing characteristics.

For pre-defined schematic attributes supported by FPGA and MachXO devices, see “List of Schematic Attributes” on page 284.

For more information on attribute usage, see the sysIO User Guides on the Lattice Web site.

Attribute Use Attributes are primarily used to control certain aspects of design optimization and signal placement. You can also use attributes to control design implementation.

Attribute Types There are two types of attributes used in Schematic Editor: *symbol* and *net*.

- **Symbol** attributes describe features related to a whole symbol. Symbol attributes usually apply only to the symbol on which they appear.
- **Net** attributes describe characteristics associated with nets.

Setting Attribute Values In the FPGA device flow, Schematic Editor passes attribute information to the output netlist as Verilog HDL or VHDL attributes depending on the project type specified when you define a project. For those attributes that can be edited, you can set or override the values as follows.

- You can set *symbol* and *pin* attribute values for all occurrences of a symbol in Symbol Editor.
- You can override the attribute values in any schematic where the symbol appears by using Schematic Editor.

Default Values Attribute values in a symbol definition become the default values for each symbol instance. These values are frequently overridden in the completed design to reflect the design's application.

When you are in Symbol Editor, any attribute values you set in Symbol Editor will be used.

When you are in Schematic Editor, any attribute values you set in Schematic Editor will be used. If you did not specify attributes for the symbol in Schematic Editor, Schematic Editor will use the values set for the symbol in Symbol Editor.

Attributes that apply to all instances of a symbol are generally assigned values when the symbol is created. Attributes that apply to a single instance (such as the instance name) are assigned after a symbol has been placed in the design.

The symbol libraries supplied with the Diamond software have predefined values for all the attributes required by most simulators and netlisters.

Displaying Attribute Values on a Schematic Attribute values are displayed in attribute windows. Before an attribute can be displayed on a schematic, an attribute window number must be assigned to the attribute in Symbol Editor.

List of Schematic Attributes

Schematic Editor for FPGA and MachXO Crossover Programmable devices supports the ability to specify HDL attributes as schematic symbol or net attributes. HDL attributes are interpreted by place and route implementation software.

Schematic Editor supports schematic attributes summarized in the following tables for Precision RTL Synthesis, Synplify Pro, and Diamond. For detailed

information on the above Synplify attributes, see “Synthesis Attributes and Directives” in the Synplify and Synplify Pro for Lattice Reference Manual.

Table 8: Table of FPGA HDL Attributes for Precision RTL Synthesis

HDL Attribute	Attribute Type	Description and Syntax
black_box	Symbol	Tells Precision RTL Synthesis that the module should not be compiled or optimized. Only the interface need be defined for synthesis. Values: True | False
don’t_touch	Symbol	Tells Precision RTL Synthesis to pass the module through synthesis without optimizing or unmapping. Values: True | False
hierarchy	Symbol	Tells Precision RTL Synthesis to maintain the hierarchy of the module. Values: Preserve | Flatten
noopt	Symbol	Tells Precision RTL Synthesis that an instance should not be optimized or changed. Values: True | False
nopad	Net	Tells Precision RTL Synthesis to not place an I/O pad on the specified port when the design is mapped to the technology. Values: True | False
preserve_driver	Net	Tells Precision RTL Synthesis to preserve the specified signal and the driver in the design. Specifies that both a signal and the signal name must survive synthesis. Values: True | False
preserve_signal	Net	Tells Precision RTL Synthesis to preserve the specified signal in the design. Values: True | False

Table 9: Table of FPGA HDL Attributes for Synplify Pro

HDL Attribute	Attribute Type	Description and Syntax
syn_black_box	Symbol	Tells Synplify Pro to treat the module as a black box during synthesis. Values: 1 | 0; True | False
syn_direct_enable	Net	Tells Synplify Pro to use the signal as the enable input to an enable flip-flop when multiple candidates are possible. (LatticeECP/EC, LatticeECP2/M, LatticeSC/M, LatticeXP, and MachXO only) Values: 1 | 0; True | False
syn_force_pads	Net	Tells Synplify Pro to enable or disable I/O insertion on a port level or a global level. Values: 1 | 0; True | False

Table 9: Table of FPGA HDL Attributes for Synplify Pro

HDL Attribute	Attribute Type	Description and Syntax
syn_hier	Symbol	Tells Synplify Pro how to control the amount of hierarchical transformation that occurs across boundaries on module or component instances during optimization. Values: Soft (default) | Firm | Hard | Remove | Macro | Flatten | Flatten,Soft | Flatten,Firm | Flatten,Remove
syn_keep	Net	Tells Synplify Pro to not remove the internal signal during synthesis and optimization. Values: 1 | 0; True | False
syn_netlist_hierarc	Symbol	Tells Synplify Pro whether or not to generate the hierarchy in the EDIF output when you assign the attribute to the top-level module of your design. The default (true) is to allow hierarchy generation. Values: 1 | 0; True | False
syn_noarrayports	Symbol	Tells Synplify Pro to treat the ports of a design unit as individual signals (scalars), not as buses (arrays) in the EDIF file. Values: 1 | 0; True | False
syn_noclockbuf	Net	Tells Synplify Pro to disable automatic clock buffer insertion. Values: 1 | 0; True | False
syn_noprune	Symbol	Tells Synplify Pro whether or not to remove instances that have outputs that are not driven. Values: 1 | 0; True | False
syn_maxfan	Net	Tells Synplify Pro the fanout limit for an individual input port or register output. (LatticeECP/EC, LatticeECP2/M, LatticeSC/M, LatticeXP, and MachXO only) Values: *integer*
syn_preserve	Net	Tells Synplify Pro to prevent sequential optimizations across a flip-flop boundary during optimization and preserves the signal. Values: 1 | 0; True | False
syn_useioff	Net	Tells Synplify Pro to pack flip-flops in the I/O ring to improve input/output path timing. (LatticeECP/EC, LatticeECP2/M, LatticeSC/M, LatticeXP, and MachXO only) Values: 1 | 0; True | False

Table 10: Table of FPGA HDL Attributes for Diamond

HDL Attribute	Attribute Type	Values
BBOX	Symbol	*<height>.<width>*
BBOXTYPE	Symbol	Obsolete. Do not use.
BLOCKNET	Net	*<name>*
BUS	Net	*<bus_name>*
COARSE	Symbol	CDEL0 | CDEL1 | CDEL2 | CDEL3

Table 10: Table of FPGA HDL Attributes for Diamond

HDL Attribute	Attribute Type	Values
COMP	Symbol	<*comp_name*>
DCSMODE	Symbol	NEG | POS | HIGH_LOW | HIGH_HIGH | LOW_LOW | LOW_HIGH | CLK0 | CLK1
DELAYTYPE	Symbol	CFGBIT | DLL | PCLK | ECLK
DIFFCURRENT	Symbol/Net	NA | 2 | 3.5 | 4 | 6
DIFFRESISTOR	Symbol/Net	Off (default) | 100 | 150 | 200 (only for differential buffers)
DIN	Symbol	<*cell_name*>
DOUT	Symbol	<*cell_name*>
DRIVE	Net	NA | 2 | 4 | 8 (default) | 12 | 16 | 24
ENDPOINT	Symbol/Net	Obsolete. Use Spreadsheet View to apply the TO keyword in the BLOCK, MAXDELAY, and MULTICYCLE constraints instead.
FINE	Symbol	FDEL0 | FDEL1 | FDEL2 | ... | FDEL47
FIXEDDELAY	Net	False | True
FREQUENCY	Net	<*value*> (expressed in MHz)
GSR	Symbol	ENABLE | DISABLE | FORCEENABLE | IPENABLE
HGROUP	Symbol	<*identifier*>
HULOC	Symbol	R<*row_number*>C<*column_number*>
HURLOC	Symbol	R<*row_number*>C<*column_number*>
IMPEDANCE	Symbol/Net	Off (default) | 25 | 33 | 50 | 100
IMPEDANCEGND	Symbol	Off (default) | 16.7 | 20 | 25 | 33 | 50 | 100 (Impedance down)
IMPEDANCEVCCIO	Symbol	Off (default) | 16.7 | 20 | 25 | 33 | 50 | 100 (Impedance up)
INBUF	Symbol	Off | On
IO_TYPE	Net	<*buffer_type*>
LOAD	Symbol	<*integer*>.<*integer*>
LOC	Symbol/Net	<*string*> Default is None (floating).
MAXDELAY	Net	<*value*> (expressed in ns)
NOCLIP	Net	1
NOMERGE	Net	<*net_name*>
NORETIME	Symbol	<*string*>
OPENDRAIN	Net	On | Off (default)
PCICLAMP	Net	On | Off (default)
PERIOD	Net	<*value*> (expressed in ps)

Table 10: Table of FPGA HDL Attributes for Diamond

HDL Attribute	Attribute Type	Values
POWERSAVE	Net	Obsolete. Do not use.
PULLMODE	Net	Up (default) | Down | None | Keeper | PCIClamp
PWRSAVE	Symbol	Off (default) | On
RBBOX	Symbol	*<height>.<width>*
REFCIRCUIT	Symbol/Net	Off (default) | Internal | External (for differential buffer)
REGION	Symbol	*<identifier>*
SLEW	Symbol	1-100
SLEWRATE	Net	Fast | Slow | NA (LatticeSC/M only)
STARTPOINT	Net	Obsolete. Use Spreadsheet View to apply the FROM keyword in the BLOCK, CLOCK_TO_OUT, MAXDELAY, and MULTICYCLE constraints instead.
TERMINATEGND	Symbol/Net	Off (default) | 50 | 100 | 120
TERMINATEVCCIO	Symbol/Net	Off (default) | 50 | 100 | 120
TERMINATEVTT	Symbol/Net	Off (default) | 60 | 75 | 120 | 150 | 210
UGROUP	Symbol	*<identifier>*
VCMT	Symbol/Net	OFF (default) | VCMT | VTT | DDR_II

Assigning Attribute Values in a Schematic

Schematic Editor enables you to assign attribute values for symbols and nets. See "List of Schematic Attributes" on page 284 for schematic attributes supported by Lattice FPGA devices.

Following the steps below, you can assign attribute values in a schematic (the procedure takes assigning the DOUT symbol attribute as an example).

To assign an attribute value in a schematic:

1. In Schematic Editor, open the schematic you want to work with.
2. In the Schematic area, select the symbol for which you want to assign the DOUT attribute.
3. Choose **Edit > Object Properties**.
4. In the pop-up dialog box, click the plus sign before **Symbols** to show the symbol name.
5. Select the symbol name. All its symbol attributes are displayed in the right-hand table.
6. In the attributes list, double-click the Value cell for **DOUT**, type **1** in the cell, and then click **Apply** to save the assignment.

   The value 1 is assigned to the DOUT attribute.

7. Click **OK** to close the dialog box.

The above example takes assigning symbol attribute values as an example. If you want to assign a net attribute value, select a net in Step 2, and then perform the same operations to show the net attributes list and assign attribute values.

The net attribute LOC is used to lock pins. Double-click the Value cell for **LOC** in the net attributes list, type in a pin name in the edit box, and then click **Apply**. The selected net will be locked to the pin.

Net attributes also include I/O configuration attributes such as IO_Type, Drive, Slewrate, Pullmode, PCIClamp, OPENDRAIN, and various synthesis tool attributes. See “List of Schematic Attributes” on page 284 for schematic attributes supported by Lattice FPGA devices.

Schematic Sheets

A sheet is a page of a schematic. A schematic file can consist multiple sheets. The schematic sheets can be added, deleted, resized, and renumbered.

Adding a New Sheet

You can add new (blank) sheets to an existing schematic.

To add a new sheet to a schematic:

1. Choose **Edit > Sheet**.
2. In the Sheet Operating dialog box, click **New**.
3. In the pop-up dialog box, enter a sheet number, select a sheet size, and click **OK**.
4. In the Sheet Operating dialog box, click **OK**.

   Schematic Editor adds the new sheet to the list.

Resizing Sheets

You can change the sheet size to any of several sizes that are based on the standard paper sizes of the American National Standards Institute (ANSI).

Table 11: Schematic Sheet Sizes

Paper Size	In Inches	In Millimeters	Similar ISO Size
ANSI A (letter)	11 x 8.5	279 x 216	A4
ANSI B (ledger or tabloid)	17 x 11	432 x 279	A3
ANSI C	22 x 17	559 x 432	A2

Table 11: Schematic Sheet Sizes

Paper Size	In Inches	In Millimeters	Similar ISO Size
ANSI D	34 x 22	864 x 559	A1
ANSI E	44 x 34	1118 x 864	A0

To resize a sheet:

1. Choose **Edit > Sheet**.
2. In the Sheet Operating dialog box, select the Size cell of the sheet that you want to change.
3. Click **Resize**.

   The Resize Sheet dialog box opens, displaying the current size of the selected sheet.
4. Select the desired new size from the drop-down menu.
5. Click **OK**.
6. In the Sheet Operating dialog box, click **OK**.

Renumbering Sheets

You can change the number of any sheet in the schematic.

To renumber a sheet:

1. Choose **Edit > Sheet**.
2. In the Sheet Operating dialog box, select the sheet number that you want to change, and click **Renumber**.

   The Change Sheet Number dialog box opens.
3. Type the new sheet number in the dialog box and click **OK**.
4. In the Sheet Operating dialog box, click **OK**.

   Schematic Editor changes the number of the sheet to the new number.

Re-sequencing Sheets

You can use the Resequence Sheet command to sequentially renumber, starting at sheet 1, all the sheets in the drawing and remove any skipped numbers.

To resequence sheets in your schematic:

1. Choose **Edit > Sheet**.
2. In the Sheet Operating dialog box, click **Resequence**, and then click **OK**.

   Schematic Editor resequences the sheets and deletes all unused (skipped) numbers.

Printing Sheets

For schematic sheets, the Print command prints whatever is showing in the Schematic Editor window. Print acts much like a screen capture tool. You may need to experiment with Print Preview to get the printout that you want.

To print schematic sheets:

1. Click to detach Schematic Editor from Diamond's main window.
2. Click **View >** **Zoom Fit**.
3. Adjust the height or width of the Schematic Editor window so the sheet fills the window with the same small margin on all sides.
4. Choose **File >** **Print Preview**.

   The Print Preview window opens.
5. Expand the Print Preview window until you are comfortable with the display of the page.
6. To maximize the printout, click for landscape mode.
7. Click the **Page Setup** button and adjust the paper size and margins if necessary.
8. If you are not happy with the way the page looks, close Print Preview, adjust the Schematic Editor window, and open Print Preview again.
9. Click the **Print** button.
10. Adjust the printer settings if necessary and click **Print**.

With a little experience, you will be able to skip using Print Preview and just choose **File >** **Print** after adjusting the Schematic Editor window.

Wiring a Schematic

Wires electrically connect schematic symbols. The symbol pins are the connection points for the wires. The Add Wire command is used to add wires between symbol pins. However, you can also use the Add Net Name and Add Bus Tap commands to add single wire segments.

Wiring Constraints

Schematic Editor enforces a number of wiring constraints. Most are intended to encourage clean layout and prevent ambiguous wiring patterns.

- All wire segments must end on Primary grid points.
- Wire segments must be oriented on the 90° axes.
- A maximum of three wires can connect at a pin or I/O port.
- A net can have only one name (simple or compound).
- Two I/O ports with different net names cannot be connected by a wire.

- An I/O port can connect only to a wire segment, never a pin. (Add a wire segment if you want a marker to be near a pin.)
- An I/O port cannot be placed in the middle of a diagonal line, only at the end.
- An I/O port cannot be placed at the crossing point of two wires, even if the wires are connected.
- A tap can only be placed on a vertical or horizontal section of a wire.
- Only one bus tap can be made at any point on a bus.
- A bus can contain only individual signals, not other buses. Attempting to give a net in a bus either a compound name or the name of another bus is flagged as an error.
- The relationship between an ordered bus (that is, a bus with a compound name) and the signals in that bus is strictly enforced. Naming the bus or one of its signals in a way that breaks this relationship is not permitted. For example, you cannot assign a bus tap a name that is not in the bus.

Drawing a Wire Using the Point-to-point Method

To draw a wire using the point-to-point method:

1. Choose **Edit > Add > Wire**.
2. Click the first point of the wire. Two dotted lines, horizontal and vertical, stretch from the selected point to the cursor. As you move the cursor, the lines change from horizontal to vertical (or vertical to horizontal) depending on the position of the cursor relative to the first point.
3. When the first dotted section has the length and direction you want, click a second time.

   The first dotted section changes to the wire color and its position is fixed. A new dotted section is now added to the “rubber band” line. You can add additional wire segments by repeating steps 2 and 3.
4. To end the wire, click twice at the same point (or right-click).

   The wire terminates automatically if the point you click falls on a pin terminal.

Drawing a 4-way Connection

To form an intersecting connection:

1. Start with a single wire.
2. Add one or two additional wires at right angles to the first wire from a point on the first wire. This type of connection is automatically marked with a connect dot to distinguish it from unconnected crossing wires.

Selecting Wires

You can select one or more wires in Schematic Editor. The selected wires are highlighted in red.

To select wires:

1. Choose **Edit > Add > Select**, or click on the drawing toolbar.
2. In the schematic, do either of the following:
   - Click a wire to select it. You can continue to select more wires by holding down the Shift key.
   - Draw a rectangle area to enclose the desired wires. The wires inside the area are selected.

Note

- When selecting wires, the area-selecting method only allows you to select one area. You cannot use the Shift key to select additional areas.
- You cannot deselect a wire by clicking the wire again. To deselect wires, click any empty space on the schematic sheet.

Changing Wire Length or Direction

After wires are placed to the schematic, you can drag the wire ends to lengthen or shorten them, or change their routing directions.

To change wire length or direction:

1. Choose **Edit > Add > Select**, or click on the drawing toolbar.
2. Choose **Edit > Drag**.
3. Select one or more wires that you want to change.
4. Place the mouse cursor near the wire end of the selected wire(s). When the cursor changes to , click and drag the wire end to the desired place.

Note

- If the node connected with the wire end that you drag only has one linked connection, dragging the wire end moves the entire wire instead of changing its length or direction.
- You cannot shorten wires if you have selected too many wires which are very close to each other. Shortening wires by dragging requires enough space to avoid wire overlapping. In such cases, you can shorten the wires one by one or adjust their positions before shortening them.

Deleting a Wire Connection

You can delete wires from the schematic in two ways using the Clear command.

To delete a single wire or wire segment:

1. Click the wire or wire segment that you want to delete.
2. Choose **Edit > Clear**.

   The wire is deleted. If the cursor points at more than one item, the items are deleted based on a priority list.

To delete wires enclosed in an area:

1. Drag a rectangle area to enclose the wires that you want to delete.
2. Choose **Edit > Clear**.

   All totally enclosed wires are deleted when you release the mouse. Portions of wires enclosed by the rectangle are also deleted. All wire segments outside the rectangle remain. All wires that cross the rectangle are cut into wire segments.

Nets and Buses

This section introduces the concepts of nets and buses, as well as procedures on creating and naming nets and buses.

Nets

Any single- or multi-wire connection between pins is called a *net* (“network”). The following topics explain how nets are named and how multi-wire nets (called *buses*) are created and named.

Net Names

The nets (networks) form the electrical connections among the components. Every net has a name, either assigned by you or by Schematic Editor. Net names have two principal functions: *identification* and *interconnection*.

Note

If your instance names use brackets **[ ]**, parentheses **( )**, or curly braces **{ },** the software replaces these characters with an underscore (_) because they are illegal characters in HDL.

Meaningful net identifiers make a design easier to understand. Nets are usually given the names of the signals they carry.

If you do not assign a name, Schematic Editor automatically assigns a unique name when you save the file, in the form of N_*nn*, where *nn* is an integer.

You can override any name assigned by Schematic Editor and assign one of your own by using the Add Net Name command from the Edit menu.

Interconnection If a wire segment attached to a symbol pin is given the name of a net or bus, the pin is attached to that net or bus, even if you haven't drawn the connection on the schematic.

Two or more wires with no visible connection on the schematic are automatically connected if they have the same net name. Each wire is called a *branch* of that net. Inter-sheet connections are created in this way.

You can easily find implicit net connections with the Drc Check command from the Design menu. Click any net or bus. All wires with the same name are highlighted, on all sheets of the current schematic.

Nets with different names cannot be connected. Schematic Editor will warn you if you try to “short” them.

Legal Characters in Net Names The following characters can be used in net names:

Table 12:

Character	Description
A–Z, a–z, 0–9	All alphanumeric characters. Case is not significant.
'	Apostrophe (single quote)
_	Underscore

Reserved Names If B is the first character of a net name, the underscore cannot follow it as the second character (as in B_). The underscore cannot be used in a net name of the form “N_*nn”* (where “*nn”* is any integer). These names are reserved by Schematic Editor for nets that have not been named by the user.

Entering a Net Name

Use the Add Net Name command from the Edit menu to assign a name of your own choice to a net. Your name replaces any name Schematic Editor may already have assigned. If you assign the same name to two separate nets (“branches”), *they are connected*, even though no connection appears on the schematic. This feature makes it easy to connect widely spaced components without having to draw long wires across the schematic.

Net names you assign are always displayed; Editor-assigned names are not displayed. To avoid cluttering the schematic, you should name only these nets:

- That connect to other schematics.
- Whose functions need documentation or clarification.
- Whose signals you want to reference in simulation or timing analysis.
- Whose signals you want to identify in the Fitter report.

To enter a net name:

1. In Schematic Editor, choose **Edit > Add > Net Name**.
2. In the pop-up dialog box, enter a name. The name is attached to the cursor. You can then place it to the schematic.

Placing a Net Name

Once a net name (or group of names) is attached to the cursor, there are three ways to place it.

- **Click an empty space** to place the net name at an empty point on a sheet. This will be an error unless a net is eventually connected to the name flag.
- **Click a net** to place the net name on the selected net. If you click at the end of a net, the net name extends from the end. If you click the middle of a net, the net name is centered just above the net.
- **Drag to create the net** and place the net name on the newly created net. The net name and a single net segment can be placed *simultaneously*. You can then place subsequent wires and names by clicking a pin.

Note

- You can only define a horizontal or vertical net segment in this way. If either end of the segment connects to a perpendicular net or bus, a bus tap is created at that end of the segment. If the net was not a bus, it is promoted to one.
- The position of the name is determined by the segment ends at the time of placement. If both ends are connected, the name is placed in the middle. If neither end of the segment is connected, the name is placed at the starting point. If only one end of the segment is connected, the name is placed at the unconnected end.
- After you drag the mouse to or from a pin, you can place subsequent net names by clicking a pin**.** You do not need to drag. A wire segment is automatically added, of the same length as the one previously dragged. The name is attached as described above.

The segment ends determine the position of the name at the time of placement. If both ends are connected, the name is placed in the middle. If neither end of the segment is connected, the name is placed at the starting point. If only one end of the segment is connected, the name is placed at the unconnected end.

After you drag the mouse to or from a pin, you can place subsequent net names by clicking a pin. You do not need to drag. A wire segment of the same length as previously dragged is automatically added. The name is attached as described above.

Renaming a Net

To rename a net:

1. In Schematic Editor, choose **Edit > Add > Net Name**.
2. In the pop-up dialog box, type the new net name. The new name is attached to the mouse cursor.
3. To rename the net (that is, all branches of the net, across all sheets), press and hold down **Shift** as you click. You can click anywhere on the net.

   The new net name replaces the old one.

   If the renamed net has an I/O port:

   - The I/O port is removed if you click to rename a specific branch.
   - The I/O port is kept if you hold down **Shift** to rename all branches.

Important

Remember that renaming a single branch disconnects that branch from the rest of the net and connects it to the net (if any) with the new name.

Note

If the name of a branch is displayed two or more times on a single branch, you cannot rename the branch. You must first use the Clear command to remove the extra name(s).

Specifying Signal Direction

An I/O port is a special indicator that identifies a net name as a device input, output, or bidirectional signal. This establishes *net polarity* (direction of signal flow) and indicates that the net is externally accessible.

The Schematic Editor Drc Check command uses I/O ports to flag any discrepancies in the polarity of marked signals and the symbol pins.

To add a port:

1. Make sure the net has a name. Nets must be named before ports can be attached.
2. Choose **Edit > Add > IO Port**.

   The Set IO Type dialog box appears.
3. In the drop-down menu on the left of the dialog box, choose the port direction. (Or select **None** to remove an existing port.)
4. If the net name is not at the end of the net where the port will go, choose **Edit > Add > Net Name**. The Set IO Type dialog box changes to the Set Wire Name dialog box. In the text box, type the name of the net. The port must be given the same name as the net.

5. Click the end of the wire. Or you can draw a box around one or more net names if the names are attached to the ends of their wires.

   The port, along with the net name, appears extending from the end of the wire.

Note

- You can place, remove, or change several ports at one time by dragging a box around the wire ends.
- A net should only have one I/O port per sheet.

To change a port:

1. Choose **Edit > Add >** **IO Port**.

   The Set IO Type dialog box appears.

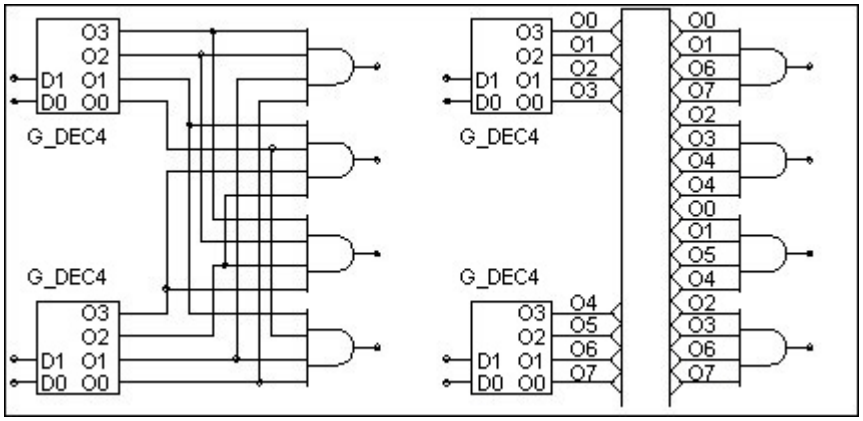
2. In the drop-down menu on the left of the dialog box, choose the new port direction. (Or select **None** to remove an existing port.)
3. Click the end of the wire. Or you can draw a box around one or more ports.

Buses

A *bus* combines two or more signals into a single wire. Buses are a convenient way to group related signals. This grouping can produce a less cluttered, functionally clearer drawing and clarify the connection between the main circuit and a Block symbol.

The figure below shows how a circuit appears before and after a bus has replaced individual wires. The two schematics are electrically equivalent.

Circuit before and after a bus has replaced individual wires.



Ordered Buses An ordered bus has a compound name consisting of the names of the signals that comprise the bus. Any signals can be combined into an ordered bus, whether they are related.

A net becomes an ordered bus when it is given a compound name. The net is *promoted* to an ordered bus containing the nets listed in the compound name. (The net is redrawn at twice its regular thickness to indicate that it's now a bus.)

A compound name is a list of two or more net names, separated by commas. For example:

```
READ,WRITE,MYNAME
```

represents the three signals READ, WRITE, and MYNAME. Spaces in compound names are ignored.

Adding a sequence of numbers to a name can also form a compound name. The sequence is specified as a starting number, an ending number, and an optional increment (default = 1). The numbers are positive integers, and are delimited by commas (**,**), dashes (**–**), or colons (**:**). The sequence is enclosed in brackets **[ ]**, parentheses **( )**, or curly braces **{ }**.

The following are examples of sequential compound names.

Table 13:

Sequential Name	Signals
DATA[0-7]	DATA[0] DATA[1] ... DATA[7]
ADDR(0,14,2)	ADDR(0) ADDR(2) ADDR(4) ... ADDR(14)
IO{4:23:3}	IO{4} IO{7} IO{10} ... IO{22}

If the increment is greater than one, the ending number will not appear in the sequence if it does not equal the starting number plus an integral multiple of the increment (as in the third example above).

A compound name can also combine individual names and compound names in any order.

Table 14:

Sequential Name	Signals
CS,DATA{0:7},WR	CS DATA{0} DATA{1} ... DATA{7} WR

The order of the signals in the bus is the same as the order in which they are specified. The order is significant only when the bus is connected to a bus pin. (Bus pins are described in a later section, ""Bus Pins" on page 301.")

Note

If your instance names use brackets **[ ]**, parentheses **( )**, or curly braces **{ },** the software replaces these characters with an underscore (_) because they are illegal characters in HDL language.

Unordered Buses An unordered bus is nothing more than an unnamed wire with *bus taps*. A net with a single name (or any unnamed wire) is promoted to an unordered bus by attaching one or more bus taps to it. The order of the signals within an unordered bus is not defined and has no significance.

Although the order of the signals in an unordered bus has no significance, *you must name the wires connecting to the bus taps*, because Schematic Editor would otherwise have no way of determining which symbol pin at one end connects to which symbol pin at the other end

Unordered buses provide a convenient way to route signals through the schematic with a minimum of visual clutter. They have no other function.

Unordered buses cannot connect to bus pins, because bus pins represent an ordered sequence of signals.

Adding a Bus Tap

Signals enter (or exit) a bus at points called *bus taps*. A bus tap can be added to any existing bus, net, or wire. If a net or wire is not already a bus, adding the tap automatically promotes it to a bus.

To add a bus tap:

1. Choose **Edit > Add > Bus Tap**.
2. Position the cursor on the bus or wire where the tap is required.
3. Drag the mouse to draw a wire perpendicular to the bus.
4. Release the mouse button when the wire is the desired length.

Bus taps can be made only on vertical or horizontal sections of a bus. Tap connections are shown as two diagonal lines.

More than one tap can be taken from the same signal, to simplify routing or permit a cleaner layout. But you cannot add a bus tap to an existing bus tap. You will get the error message "Forming Multilevel Bus."

Naming the Tap

Once the tap has been added, choose the **Edit > Add > Net Name** command to name it. If the tap is from an ordered bus, the tap's name must match the name of a signal in the bus. If it does not, Schematic Editor will flag it as an error.

Note

Wires entering and leaving any bus (ordered or unordered) must be tagged with a net name to indicate which signal is being tapped. Unnamed taps will eventually be flagged as errors.

Bus Pins

A *pin* represents either a physical pin on a real component or a signal from a lower-level schematic.

A *bus pin* represents a group of pins or signals. You create a bus pin by giving a pin a compound name. That is, a list of signals. If the pin connects to a Block symbol, each of the signals listed in the bus pin name must also appear in the schematic. This defines the connection between the Block symbol and its underlying schematic.

Ordered buses can connect directly to bus pins. The number of bits or signals attached to the bus must match the number of bits or signals attached to the bus pin.

The first signal in the bus (by definition, the first signal in the bus's name) is connected to the first signal represented by the pin. The remaining signals in the bus are connected to the remaining pins in the same order you assigned the signal names to the pins.

Connecting to Pins

A tapped signal connects to an ordinary symbol pin in the usual way. An ordered bus connects to a *bus pin* (a pin with multiple connections) directly. No taps are needed; the connections are made automatically. The first signal in the bus connects to the first signal in the bus pin, the second to the second, and so forth. Both the bus and the bus pin must contain the same number of signals.

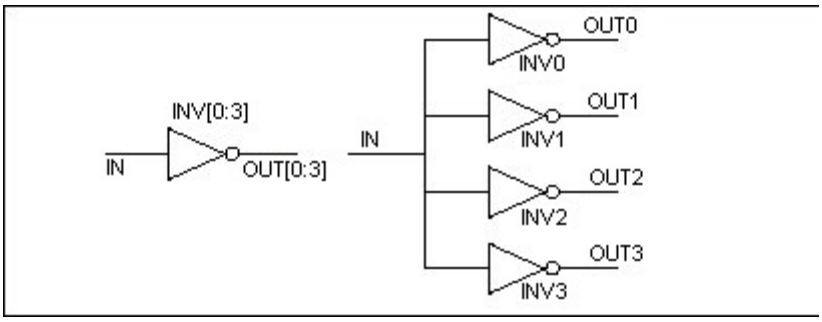
Compound Names

The input and output nets of an iterated instance can be given either single names or compound names. If the inputs or outputs are given a compound name, their nets are promoted to buses in which each instance's input or output is a separate signal.

Iterated buses work like any other bus. You can attach a bus (with the same number of signals) directly to them, as you would any other bus.

Single Names

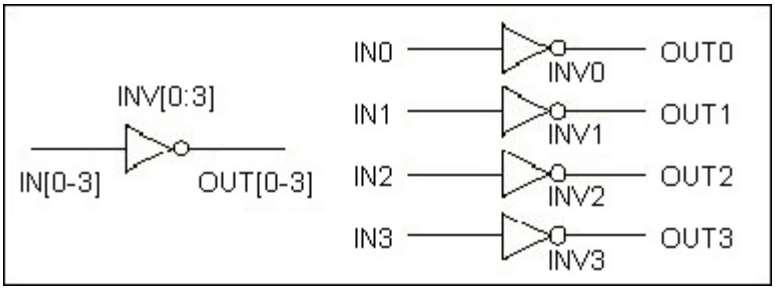
If an iterated input or output is given a single net name, there is only one input or output net, and all the inputs or outputs connect in parallel to that single net. In the figure below, the input net is given a single name and the inputs of all four gates are connected in parallel to the net.



This feature is used most often on inputs, not outputs. Paralleled outputs represent a “wired-OR” configuration, which is usually drawn as separate gates, rather than as an iterated instance.

Nets on Iterated Instances

Iterated instances allow a single symbol instance to represent multiple instances in a parallel connection. The figure below shows two ways of representing four parallel buffers. On the right, four separate inverters are added to the schematic. On the left, one symbol with the instance name of INV[0:3] represents the bank of four inverters.



Bus and Net Connections to Iterated Instance

You can make an iterated instance of any symbol. A simple (non-bus) pin on an iterated instance remains a simple pin. The iteration does not convert the pin to a bus pin. The same rules of connection to a simple pin still apply.

Connections to the nets of iterated instances are made according to the following rules.

Table 15: Rules for Connection to Nets of Iterated Instances

Connection	Rule
Simple net to simple pin	The net connects to corresponding pin on each instance.
Bus to bus pin	Successive bus signals connect to successive bus pins on successive instances. The bus and bus pin must have the same number of nets.
Simple net to bus pin	Not permitted. Only a bus can connect to a bus pin.
Bus to simple pin	The Nth signal of the bus connects to the scalar pin of the Nth instance. The width of the bus and the number of instances must match.

Creating a Tap, Wire, and Net Name At Once

You can use the Net Name command in Schematic Editor to create the tap, a wire, and the net name at once.

To create a tap using the Net Name command:

1. In Schematic Editor, choose **Edit > Add > Net Name**.
2. In the pop-up dialog box, type a net name contained in the bus to be tapped. The name is attached to the mouse cursor.
3. Point to the place on the schematic where you want the free end of the bus tap.
4. Drag the mouse to the bus and release the mouse button.

   The tap is made, and the free end is labeled with the signal name.

Saving a Schematic

The Save File command in Schematic Editor saves the changes to the existing schematic file. You can use the Save File As command to save the schematic with another file name.

Schematic files have the extension .sch. Below shows the steps of saving a schematic with the Save File As command.

To save a schematic with the Save File As command:

1. In Schematic Editor, choose **File > Save File As**.

   The Save as Schematic File dialog box opens.
2. Go to the folder in which you want to save the schematic.
3. Type in a base name for the schematic.

   Schematic Editor adds the .sch extension automatically when you specify the base name.

4. Click **Save**.

Debugging a Schematic

Schematic Editor has two levels of checking that report or prevent errors early in the design process.

First level errors are detected as you enter your schematic. For example, Schematic Editor will not let you draw an isolated wire that forms a closed loop without connecting to anything else. It won't let you short together nets with different names.

Second level errors are recognized in the context of a complete design. An unconnected wire or pin, or an unnamed signal tapped from a bus, are normal during the first stages of a design. Some potential errors are always indicated, such as the dots on open pins and hanging line ends. Otherwise, errors of this type are reported only when schematics and symbols are combined in the design hierarchy.

You can check for errors and potential errors at any time.

See Also ▶"Checking a Schematic for Errors" on page 304

Checking a Schematic for Errors

You can use the Drc Check command to detect errors that are only meaningful in a completed schematic or symbol.

Schematic Errors The following schematic errors are detected:

- Bus Taps should be named.
- Isolated Net Name Flags are not permitted.
- If the ShowSymbolPins option is enabled, each pin on symbols in this schematic should be connected to a net.
- If the MarkOpenEnds option is enabled, there should not be any unconnected net ends.
- Only ordered buses can be connected to a bus pin on a symbol.
- Only ordered buses can be marked with I/O ports.
- Nets should not be marked with more than one I/O port.
- A bus tap and its bus should not both be marked with an I/O port.
- If there is a symbol for this schematic, the following errors are detected by the schematic checker:
- Each pin must have a corresponding net with an I/O port whose direction matches the Polarity of the pin.
- Each net marked with an I/O port must correspond to a pin.

Symbol Errors The following symbol errors are detected:

- Block symbols should have a schematic with the same name in the current directory.
- Symbols of a type other than Block are primitives and should not have a schematic of the same name.
- Each pin in a Block or Cell must have a Name. Each pin in a Gate or Component should have a PinNumber.
- Every pin in a Gate must appear in the same number of sections. Therefore, all PinNumber attributes must have the same count of pin numbers.
- Pins in Component and Pin symbols can have only one PinNumber.
- Pins in the same Gate group must have the same Polarity, Load and Drive.
- Pins on symbols that are not Blocks should have Load or Drive specifications. Input pins should have Load but not Drive. Output pins should have Drive, but not Load, unless the pin is tristate, in which case it should have a Load that represents the load in the High-Z state. Bidirectional pins should have both Load and Drive.
- Pins on Blocks should not have Load or Drive specifications.

Checking Errors

To check a schematic for errors:

- Choose **Design > Drc Check**.

  The software opens an error report in a pop-up text window and writes the information to a file.

Highlighting a Net

You can use the Select icon to trace nets in a schematic. Selecting a net changes its color (or turns it into a dashed line on a monochrome monitor). If the net is a signal in a bus, the bus is also highlighted.

To highlight a net:

1. Click on the drawing toolbar.
2. Click the net that you want to highlight.

   The color of the net changes.
3. To remove highlighting, click any blank area in the schematic.

Troubleshooting FPGA Schematic Design

This section describes common problems and solutions for the schematic design flow.

Verilog Functional Simulation Model (vericode.exe) or VHDL Functional Simulation Model (vhdl.exe):

```
ERROR: Missing I/O marker for pin <pin name>
ERROR: Missing pin for I/O marker on net <net name>
```

Caused by a mismatch between I/O ports of a schematic (.sch) and a related schematic symbol (.sym). A common mistake is to overlook creating a symbol for the top-level schematic of a project.

```
ERROR: Polarity mismatch on pin – I/O marker for net <net name>
```

Caused by a polarity (or type) mismatch between an I/O port and symbol pin. A common mistake is attempting to read or feedback a signal marked with an Output marker.

```
ERROR: --- Pins in symbol - <symbol name>.sym
ERROR: --- IOs in schematic  - <schematic name>.sym
```

Caused by a mismatch between I/O ports and a related schematic symbol. A common mistake is modify the I/O ports and not update the related schematic symbol (.sym). A common mistake is to overlook creating a symbol for the top-level schematic of a project.

Schematic Editor > DRC:

```
Different I/O marks on net <net name>
```

Caused by multiple I/O ports on the same net.

```
Pin <pin name> has wrong I/O mark
```

Caused by a mismatch between I/O ports and a related schematic symbol. A common mistake is that you modified the I/O ports and not updated the related schematic symbol (.sym). A common mistake is to overlook creating a symbol for the top-level schematic of a project.

```
Bus <bus name> and signal <signal name> marked as I/O
```

Caused by multiple I/O ports on the same net.

```
Unconnected pin <instance name>-<pin name>
```

Caused by an unwired pin. In general it is good practice to tie unused inputs to a logic constant such as the VHI or VLO library element. Unused outputs can remain unconnected.

```
Signal count mismatch on pin <instance name>-<pin name>
```

Caused by a range mismatch between the bus pin of a symbol instance and the bus wire.

ModelSim:

```
Unresolved reference to GSR_INST
Unresolved reference to PUR_INST
Unresolved reference to TSALL_INST
```

ModelSim reports:

```
** Error: (vsim-3043) <path>/cae_library/simulation/verilog/
machxo/FL1S3AY.v(15): Unresolved reference to 'PUR_INST'
```

Many synchronous library elements from the Lattice FPGA Symbol Library contain references to global signals related to Global Set/Reset (GSR), Power Up Reset (PUR), or Tri-State All (TSALL). To establish a driver on these “global” nets you must instantiate the related library element in your design or test fixture. For more information, refer to the following topics:

- “How to Use the Global Set/Reset (GSR) Signal” on page 1174
- “How to Use the Power Up Set/Reset (PUR) Global Signal” on page 1189
- “How to Use the Tristate Interface (TSALL) Global Signal” on page 1190

Customizing Work Environment

You can use the Options dialog box to set schematic display and work environment for the schematic tools.

To access Schematic Editor or Symbol Editor options, choose **Tools > Options** in the Diamond main window. Then select **Schematic Editor** or **Symbol Editor** in the Options dialog box.

Turning On/Off Grid Display

You can use the Options dialog box to turn on or off the grid display in Schematic Editor and Symbol Editor.

To turn grid display on or off:

1. In the Diamond window, choose **Tools > Options**.

   The Options dialog box opens.
2. In the dialog box, click **Graphic Options** under Schematic Editor or Symbol Editor.
3. Select **Show Grid**.
4. Click **Apply** or **OK**.

Changing the Color Scheme

You can change the current color scheme for Schematic Editor and Symbol Editor.

To change the current color scheme:

1. In the Diamond window, choose **Tools > Options**.

   The Options dialog box opens.

2. In the dialog box, select **Color Options** under Schematic Editor or Symbol Editor.
3. Select the colors you want to use as the new scheme.
4. Click **Apply** or **OK**.

Tool Options for Schematic Editor Colors

You can change the colors for the following items in the dialog box:

Background: Color of sheet background

Wire: Color of wire on sheet

Symbol: Color of symbol on sheet

Add Wire: Not used

Add Bustap: Not used

Highlight: Color of highlighted objects

Selected: Color of selected objects

Drc Check: Color of cross line which marks errors

Panel: Color of sheet panel including frame and pixels

Selected Box: Not used

Tool Options for Symbol Editor Colors

You can change the colors for the following items in the dialog box:

Background: Color of sheet background

Wire: Color of wire on sheet

Symbol: Color of symbol on sheet

Add Wire: Not used

Add Bustap: Not used

Highlight: Color of highlighted objects

Selected: Color of selected objects

Drc Check: Color of cross line which marks errors

Panel: Color of sheet panel including frame and pixels

Selected Box: Not used

Simulating the Design

This topic explains how to perform functional and timing simulation by using the Diamond tools and third-party simulators.

See Also

- "Functional Simulation" on page 311
- "Simulation Environment" on page 311
- "Design Source Files" on page 311
- "Creating a New Simulation Project in Diamond" on page 312
- "Exporting VHDL and Verilog Simulation Files" on page 313
- "Verifying Designs with Timing Simulation" on page 314
- "Timing Simulation in the FPGA Process Flow" on page 315
- "Third-Party Simulators" on page 315

Simulation in Diamond

Diamond provides you with an interface to create a new simulation project file that you can import into a standalone simulator. Diamond supports Active-HDL and ModelSim simulation file for file exports. To create testbenches for simulation in Diamond, generate all testbenches in your third-party tool using an imported simulation project (.spf) file.

See Also

- "Functional Simulation" on page 311
- "Simulation Environment" on page 311
- "Design Source Files" on page 311
- "Creating a New Simulation Project in Diamond" on page 312
- "Exporting VHDL and Verilog Simulation Files" on page 313

Functional Simulation

Functional simulation is the process of verifying that the model of the design matches the functional specification. Functional simulation helps identify logic errors in a design before it is programmed into a device.

In the Diamond design flow, to perform functional simulation on designs modeled with Verilog HDL or VHDL, you can peform pre-synthesis functional simulation on your source HDL or netlist using a third party simulator. In Diamond use the subprocesses **Verilog Simulation File** and **VHDL Simulation File** to produce a post-map source simulation file for functional simulation of primitive gate-level logic during that stage in the flow..

HDL Simulators The Diamond software and model libraries are qualified with a variety of popular HDL simulators, including Aldec Riviera Pro®, Aldec Active-HDL®, Cadence® NC-Verilog®, Cadence® NC-VHDL®, Cadence® NCSim®, Mentor Graphics® ModelSim®, Mentor Graphics Questa®,and Synopsys® VCS®.

See Also ▶"Simulation in Diamond" on page 310

Simulation Environment

In the Diamond software environment provides library model resources in both Verilog HDL and VHDL formats to support stand-alone simulation with Aldec, Cadence, Mentor Graphics, and Synopsys simulators. Using the simulation wizard inside of the Diamond software, you can create simulation project files that can be exported and brought into standalone simulators.

In addition, you can use Diamond's backannotation tools to extract gate-level models and the timing data. This type of simulation is convenient if you want to simulate a design file outside the current project. Stand-alone operation enables you to choose your own settings and preferences.

You can customize the simulation by creating a different script (.do) file that contains commands equivalent to the ModelSim or Active-HDL graphical user interface (GUI) commands. For information about creating your own simulation scripts, see ModelSim User's Manual.

See Also ▶"Simulation in Diamond" on page 310

Design Source Files

The third-party simulators enable you to simulate the operation of your design in the following design entry formats:

- Schematic format (<*design*>.sch) – Describes the circuit in terms of the components used and how they connect to each other. These schematics are automatically translated into Verilog or VHDL format.
- VHDL format (<*design*>.vhd) – Describes the circuit in Very High-Speed IC hardware description language (VHDL) format.

- Verilog HDL format (<*design*>.v) – Describes the circuit in Verilog format, which is an industry-standard hardware description language used to describe the behavior of hardware that can be implemented directly by logic synthesis tools.

See Also "Simulation in Diamond" on page 310

Creating a New Simulation Project in Diamond

This topic describes how to use the Diamond Simulation Wizard to create a simulation project (.spf) file so you can import it into a standalone simulator.

1. In Diamond, click **Tools > Simulation Wizard** or click the  icon in the toolbar. The Simulation Wizard opens.
2. In the Preparing the Simulator Interface page click **Next**.
3. In the Simulator Project Name page, enter the name of your project in the Project Name text box and browse to the file path location where you want to put your simulation project using the Project location text box and Browse button.

   When you designate a project name in this wizard page, a corresponding folder will be created in the file path you choose. If you add a subdirectory to the default file path the wizard will prompt you to create a new directory. Click **Yes** in the popup dialog that asks you if you wish to create a new folder.
4. Click either the **ModelSim** or **Active-HDL** simulator check box and click **Next**. Note that if Active-HDL is not installed, it will appear grayed out and will not be selectable.

   Note

   If you want to use Active-HDL, you must precompile all of the Lattice library files in the simulator before you proceed through the Simulation Wizard. Precompiled library files are available for ModelSim in the ./(*install_folder*)/modeltech/lib folder.
5. In the Process Stage page choose what type of Process Stage of simulation project you wish to create. Valid types are RTL, Post-Synthesis Gate-Level, Post-Map Gate-Level, and Post-Route Gate-level+Timing. Only those process stages that are available are activated.

   Note that you can make a new selection for the current Strategy if you have more than one defined in your project.

   The software supports multiple strategies per project implementation which allows you to experiment with alternative optimization options across a common set of source files. Since each strategy may have been processed to different stages, this dialog allows you to specify which stage you wish to load.
6. In the Add Source page, select from the source files listed in the Source Files list box or use the browse button on the right to choose another desired source file.

- By default, the **Analyze HDL and test bench** option is checked. This option allows you to specify a top level simulation testbench. Uncheck this box if you do not wish to specify a top level simuation testbench.
- If you wish to keep the source files in the local simulation project directory you just created, check the Copy Source to Simulation Directory option.

7. Click **Next**. If the **Analyze HDL and test bench** option was checked in the previous page, specify a name for the top level simulation test bench in the Simulation top box the Parse HTL and Test Bench page.
8. Click **Next**, and a Summary page appears and provides information on the project selections including the simulation libraries. By default the **Run simulator, Add top-level signals to waveform display, and Run simulation** check boxes are enabled. Their functionality is as follows:
   - The **Run simulator** option will launch the simulation tool you chose earlier in the wizard in the Simulator Project Name page when you perform Step 9.
   - The **Add top-level signals to waveform display** option can only be selected if the **Run simulator** box is checked. This option adds top-level signals to the waveform in the simulation tool when you perform Step 9.
   - The **Run simulation** option can only be selected if the **Add top-level signals to waveform display** option is checked. This option causes your simulation tool to run the simulation automatically when you perform Step 9.
9. Click **Finish**. The Simulation Wizard Project (.spf) file and a simulation script DO file are generated after running the wizard. You can import the DO file into your current project if desired. If you are using Active-HDL, the wizard will generate an .ado file and if you are using ModelSim, it creates and .mdo file.

See Also ▶"Simulation in Diamond" on page 310

Exporting VHDL and Verilog Simulation Files

In Diamond, you can export project simulation source files to simulate into your third-party vendor simulation tool for timing simulation.

To export a VHDL or Verilog file in Diamond:

1. In the Process view, under the **Export Files** process check either the **VHDL Simulation File** or **Verilog Simulation File** subprocess.
2. Double click the **Export Files** process to run the export feature. This will generate a back-annotated version of your design file in an .ldb file, a VHDL netlist (.vho) simulation file, and an SDF timing file.

Note:

You can also use Tcl Diamond commands in the Tcl Console to accomplish this task as shown below:

```
prj_run Export -task TimingSimFileVHD
prj_run Export -task TimingSimFileVlg
```

See Also ▶"Simulation in Diamond" on page 310

Timing Simulation

The Diamond software supports two types of timing verification: dynamic timing simulation and static timing analysis. See the Static Timing Analysis topic and related information for more information on that type of design verification.

To facilitate timing simulation, Diamond allows you to export simulation files that you can import into a simulator for design verification using dynamic timing simulation. In the Process view under **Export Files** use the **Verilog Simulation File** and **VHDL Simulation File** to export a simulation project file to a standalone Aldec Active-HDL® or Mentor Graphics® ModelSim® hardware design language (HDL) simulator. Some simulators allow a combination of both languages to be simulated concurrently. This scenario is common when mixing IP cores.

See Also ▶"Verifying Designs with Timing Simulation" on page 314

▶ "Timing Simulation in the FPGA Process Flow" on page 315

Verifying Designs with Timing Simulation

Timing simulation is based on an event-driven simulator and requires you to specify a test vector (waveform). Where timing analysis returns partial timing information, dynamic timing simulation (timing simulation) gives you detailed information about gate delays and worst-case circuit conditions. Because the total delay of a complete circuit depends on the number of gates the signal sees and on the way the gates have been placed in the device, you can only perform timing simulation after you have implemented the design. Timing simulation also requires several input files to run.

Diamond simulation files can be imported into HDL simulators (e.g., Aldec's Active-HDL or Mentor Graphics' ModelSim) for dynamic timing simulation. You can simulate FPGA devices by using any popular third-party HDL

simulator. Dynamic simulation analyzers have considerably longer run times than a static timing analysis tool like the Diamond TRACE program.

Note

You cannot generate text fixtures in Diamond. All simulation must be done outside of the Diamond software using simulation files in a supported simulator. For guidelines using third-party EDA simulators with Diamond software, see your vendor documentation or the following references.

See Also ▶"Timing Simulation" on page 314

Timing Simulation in the FPGA Process Flow

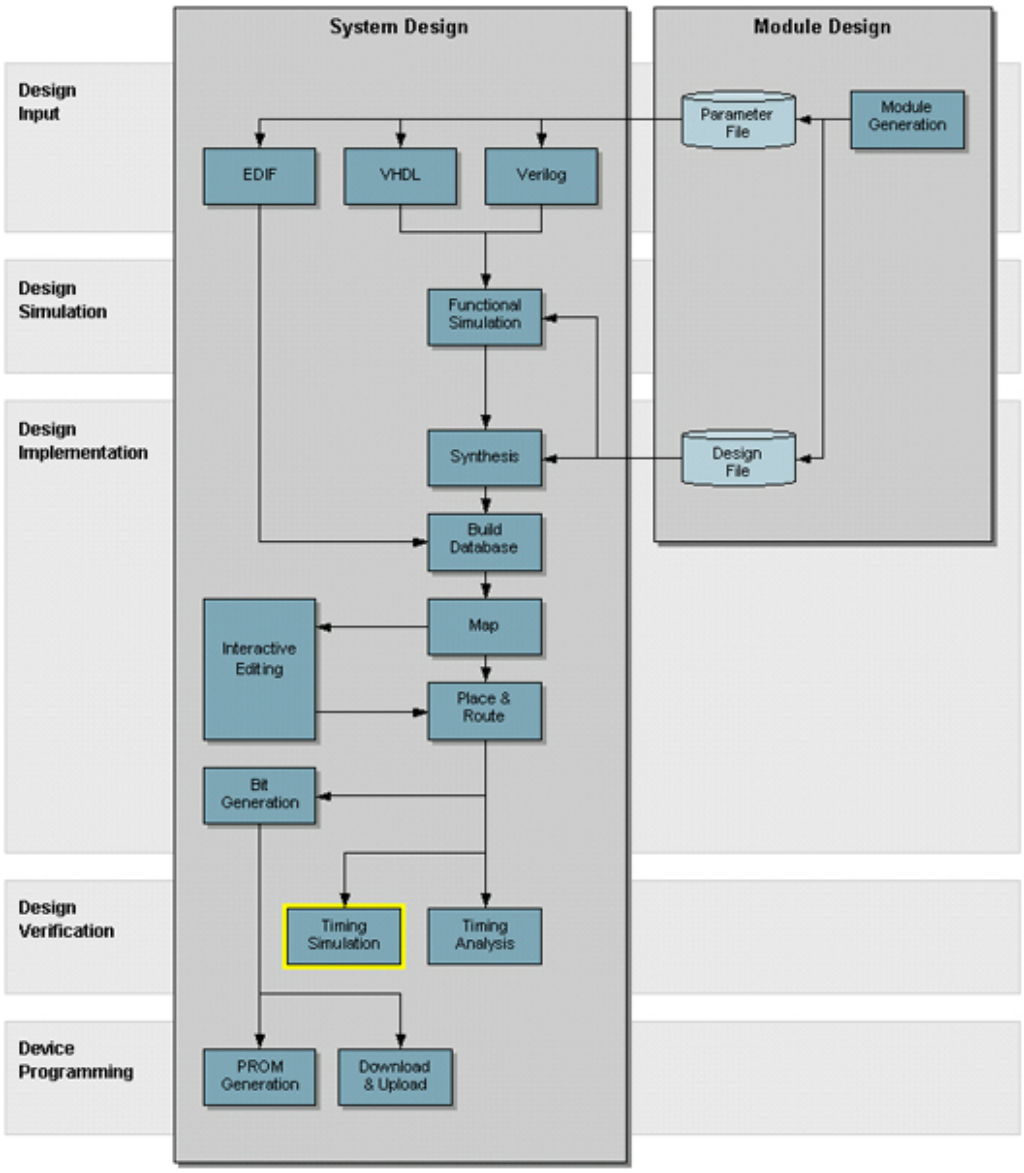
The following figure shows the place of timing simulation in the FPGA process flow as a part of design verification. To learn more about timing analysis using static timing analysis tools, see the appropriate topics in this Help system.

See Also "Timing Simulation" on page 314

Third-Party Simulators

This section explains how to use Aldec Active-HDL, Aldec Riviera Pro, Cadence NC-Verilog, Cadence NC-VHDL, Cadence NCSim, Mentor Graphics ModelSim / Questa, and Synopsys VCS software to simulate designs that target Lattice Semiconductor FPGAs. It shows you how to use these

Figure 47: Timing Simulation Flow Chart



simulators to perform functional register-transfer-level (RTL) simulation, post-map simulation, and place-and-route gate-level simulation with and without timing simulation.

Note

Lattice Semiconductor does not supply the Synopsys VCS, Cadence NC-Verilog, Cadence NC-VHDL, Cadence NCSim, Aldec Riviera Pro, or Mentor Graphics ModelSim / Questa simulators. You must obtain them independently.

See Also ▶"Simulation Library Files" on page 317

▶ "Performing Simulation with Aldec Riviera Pro" on page 319

- "Performing Simulation with Aldec Active-HDL" on page 323
- "Performing Simulation with Cadence NC-Verilog" on page 324
- "Performing Simulation with Cadence NC-VHDL" on page 329
- "Performing Mixed-Language (Verilog and VHDL) Simulation with Cadence NCSim" on page 334
- "Performing Simulation with Mentor Graphics ModelSim / Questa" on page 335
- "Performing Simulation with Synopsys VCS" on page 337
- "Performing Mixed-Language (Verilog and VHDL) Simulation with Synopsys VCS-MX" on page 338
- "Performing Mixed-Language (Verilog and SystemVerilog) Simulation with Synopsys VCS-MX" on page 339
- "Referencing Black-Box Models" on page 340

Simulation Library Files

This section describes where to find the source simulation library files for FPGA device families.

There are pre-compiled library files available for ModelSim/Quest located at (*install_folder*)/modeltech/lib.

VHDL and Verilog library source files are also installed in the cae_library directory.

The sources of some of the library models are not provided, so Lattice delivers them as black-boxes. This means for those models, Lattice provides either precompiled libraries (ModelSim / Questa) or encrypted HDL sources (Riviera Pro, NCSim, and VCS). For further details, refer to "Referencing Black-Box Models" on page 340.

Verilog Source Library Files The Verilog functional and behavioral simulation library files are installed with the Diamond software in the locations shown in the following table.

Table 16: Verilog Functional and Behavioral Simulation Library File Locations

FPGA Family	Library Location
ECP5U/UM	<*install_dir*>/cae_library/simulation/verilog/ecp5u
LatticeEC	<*install_dir*>/cae_library/simulation/verilog/ec
LatticeECP	<*install_dir*>/cae_library/simulation/verilog/ecp
LatticeECP2/M	<*install_dir*>/cae_library/simulation/verilog/ecp2
LatticeECP3	<*install_dir*>/cae_library/simulation/verilog/ecp3
LatticeXP	<*install_dir*>/cae_library/simulation/verilog/xp

Table 16: Verilog Functional and Behavioral Simulation Library File Locations

FPGA Family	Library Location
LatticeXP2	<*install_dir*>/cae_library/simulation/verilog/xp2
LatticeSC	<*install_dir*>/cae_library/simulation/verilog/sc
LatticeSCM	<*install_dir*>/cae_library/simulation/verilog/scm
LIFMD	<*install_dir*>/cae_library/simulation/verilog/lifmd
LIFMDF	<*install_dir*>/cae_library/simulation/verilog/lifmdf
LIFMNX	<*install_dir*>/cae_library/simulation/verilog/lfmnx
MachXO	<*install_dir*>/cae_library/simulation/verilog/machxo
MachXO2	<*install_dir*>/cae_library/simulation/verilog/machxo2
MachXO3D	<*install_dir*>/cae_library/simulation/verilog/machxo3d
MachXO3H	<*install_dir*>/cae_library/simulation/verilog/machxo3h
MachXO3L	<*install_dir*>/cae_library/simulation/verilog/machxo3l
Platform Manager	<*install_dir*>/cae_library/simulation/verilog/lptm
Platform Manager2	<*install_dir*>/cae_library/simulation/verilog/lptm2

VHDL Source Library Files The VHDL functional and behavioral simulation library files are installed with the Diamond software in the locations shown in the following table.

Table 17: VHDL Functional and Behavioral Simulation Library File Locations

FPGA Family	Library Location
ECP5U/UM	<*install_dir*>/cae_library/simulation/vhdl/ec/ecp5u
LatticeEC	<*install_dir*>/cae_library/simulation/vhdl/ec/src
LatticeECP	<*install_dir*>/cae_library/simulation/vhdl/ecp/src
LatticeECP2/M	<*install_dir*>/cae_library/simulation/vhdl/ecp2/src
LatticeECP3	<*install_dir*>/cae_library/simulation/vhdl/ecp3/src
LatticeXP	<*install_dir*>/cae_library/simulation/vhdl/xp/src
LatticeXP2	<*install_dir*>/cae_library/simulation/vhdl/xp2/src
LatticeSC	<*install_dir*>/cae_library/simulation/vhdl/sc/src
LatticeSCM	<*install_dir*>/cae_library/simulation/vhdl/scm/src
LIFMD	<*install_dir*>/cae_library/simulation/vhdl/scm/lifmd
LIFMDF	<*install_dir*>/cae_library/simulation/vhdl/scm/lifmdf
LIFMNX	<*install_dir*>/cae_library/simulation/vhdl/scm/lfmnx

Table 17: VHDL Functional and Behavioral Simulation Library File Locations

FPGA Family	Library Location
MachXO	<*install_dir*>/cae_library/simulation/vhdl/machxo/src
MachXO2	<*install_dir*>/cae_library/simulation/vhdl/machxo2/src
MachXO3D	<*install_dir*>/cae_library/simulation/vhdl/machxo3d
MachXO3H	<*install_dir*>/cae_library/simulation/vhdl/machxo3h
MachXO3L	<*install_dir*>/cae_library/simulation/vhdl/machxo3l
Platform Manager	<*install_dir*>/cae_library/simulation/vhdl/lptm
Platform Manager 2	<*install_dir*>/cae_library/simulation/vhdl/lptm2

See Also "Third-Party Simulators" on page 315

Performing Simulation with Aldec Riviera Pro

This section explains how to perform simulation with the Aldec Riviera Pro simulator.

Creating or Updating Lattice Semiconductor's FPGA Vendor Libraries

Create VHDL or Verilog libraries by following the procedures in this section.

Creating VHDL Libraries If you do not have Lattice Semiconductor's FPGA VHDL libraries as pre-compiled vendor libraries, the following steps are required to create them:

1. Create Lattice Semiconductor's FPGA VHDL libraries:

```
# create new VHDL libraries
alib ecp5u
alib sc
alib ec
alib xp
alib ecp
alib machxo
alib ecp2
alib xp2
alib ecp3
alib machxo2
alib machxo3d
alib machxo3l
alib lptm
alib lptm2
alib lifmd
alib lifmdf
alib lfmnx
amap scm ./sc
amap ecp2m ./ecp2
```

```
amap ecp5um ./ecp5u
```

Note

The VHDL library names are strict.

2. For each of the above VHDL libraries compile the source files by using the `acom` command:

```
acom -work <library_name> -reorder <vhdl_lib_src_folder>/
*.vhd
```

where:

- *<vhdl_lib_src_folder>* is the folder containing VHDL source files for the appropriate Lattice Semiconductor library *<library_name>*.

For the ecp5u library, there is an extra step required. You must also compile the Verilog source files by using the `alog` command:

```
alog -work <library_name> <vhdl_lib_src_folder>/*.v
```

where:

- *<vhdl_lib_src_folder>* is the folder containing VHDL source files for the appropriate Lattice Semiconductor library *<library_name>*.

If you want to place the debugging information in the library, use the –dbg switch. Debugging may slow down simulation.

3. Set the status of the newly created library as read-only to prevent accidental overwriting:

```
setlibrarymode -ro <library_name>
```

Creating Verilog Libraries If you do not have Lattice Semiconductor's FPGA Verilog libraries as pre-compiled vendor libraries, the following steps are required to create them:

1. Create Lattice Semiconductor's FPGA Verilog libraries:

```
# create new Verilog libraries
alib ovi_ecp5u
alib ovi_sc
alib ovi_ec
alib ovi_xp
alib ovi_xp2
alib ovi_ecp
alib ovi_ecp2
alib ovi_ecp3
alib ovi_machxo
alib ovi_machxo2
alib ovi_machxo3d
alib ovi_machxo3l
alib ovi_lptm
alib ovi_lptm2
alib ovi_lifmd
alib ovi_lifmdf
alib ovi_lfmnx
amap ovi_scm ./ovi_sc
```

```
amap ovi_ecp2m ./ovi_ecp2
amap ovi_ecp5um ./ovi_ecp5u
```

Note

The Verilog library names are not strict, but they conform to Aldec's naming convention for Verilog vendor libraries.

2. For each of the above Verilog libraries, compile the source files by using the alog command:

   ```
   alog -quiet -work <library_name> <verilog_lib_src_folder>/
   *.v
   ```

   where <*verilog_lib_src_folder*> is the folder containing the Verilog source files for the <*library_name*> library.

   If you want to place the debugging information in the library, use the –dbg switch. Debugging may slow down simulation.

   Note

   Ignore the warning messages.

3. Set the status of the newly created library as read-only to prevent accidental overwriting:

   ```
   setlibrarymode -ro <library_name>
   ```

Updating Lattice Semiconductor's Vendor Libraries If you obtained the updated source code of Lattice Semiconductor's FPGA libraries, you can update the VHDL libraries, Verilog libraries, or both by using the following procedure (for each one of the VHDL or Verilog libraries):

1. Using the cd command, navigate to the directory where your vendor library resides. The vendor libraries are usually located in the <*install_dir*>/vlib folder when pre-compiled by Aldec.
2. Set the library mode to read-write by using the setlibrarymode command:

   ```
   setlibrarymode -rw <library_name>
   ```

3. Compile the source code into the library. Use the acom command for VHDL source files and the alog command for Verilog source files.

   ```
   acom -work <library_name> -reorder <library_src_folder>/
   *.vhd
   ```

   or

   ```
   alog -quiet -work <library_name> <library_src_folder>/*.v
   ```

   Note

   For the ecp5u VHDL library, you must perform both of the above steps.

4. Set the library to the read-only mode to prevent accidental overwriting:

```
setlibrarymode -ro <library_name>
```

Note

You can enter the commands either in command-line mode or in the graphical console. Alternatively, you can write a macro that will prepare and compile the libraries, as follows:

1. Include in the macro the appropriate commands for the libraries that you want to create or update.
2. Run the runvsimsa script (Linux) or the runvsimsa.bat batch file (Windows) command with the macro as a command-line argument.

Performing Verilog Simulation Use the procedures described in this section to perform a Verilog simulation.

Functional RTL Simulation Use the following commands to perform a functional RTL simulation with one of the libraries shown in Table 16:

```
alib work
alog -v2k -work work <design_name>.v <test_bench>.v
asim work.<test_bench>
```

Note

If you have a pre-existing work library, you can clear its contents by using the following command:

```
adel -lib work -all
```

Post-Map and Place-and-Route Gate-Level Simulation Use the following commands to perform post-map and place-and-route gate-level simulation:

```
alib work
alog -quiet -v2k -work work -l <library_name> <design_name>.vo
<test_bench>.v
asim -O5 +access +rw -L <library_name> work.<test_bench>
```

where <*library_name*> is the FPGA Verilog library to be searched for the low-level units.

Post-Map and Place-and-Route Gate-Level Simulation with Timing Use the following procedure to perform post-map and place-and-route gate-level simulation with timing:

1. Add the `sdf_annotate` line at the top-level test bench:

   ```
   $sdf_annotate ("<design_name>.sdf", <instance_name>,
   ,<sdf_log>, "MAXIMUM");
   ```

2. Use the following commands to perform the simulation:

   ```
   alib work
   alog -quiet -v2k -work work -l <library_name>
   <design_name>.vo <test_bench>.v
   asim -O5 +access +rw -L <library_name> work.<test_bench>
   ```

Performing VHDL Simulation Use the procedures described in this section to perform a VHDL simulation.

Functional RTL Simulation Use the following commands to perform a functional RTL simulation with one of the libraries shown in Table 17:

```
alib work
acom -work work <design_name>.vhd <test_bench>.vhd
asim -t ps work.<test_bench>
```

Note

If you have a pre-existing work library, you can clear its contents by using the following command:

```
adel -lib work -all
```

Post-Map and Place-and-Route Gate-Level Simulation Use the following commands to perform post-map and place-and-route gate-level simulation:

```
alib work
acom -work work <design_name>.vho <test_bench>.vhd
asim -t ps -noglitchmsg work.<test_bench>
```

Post-Map and Place-and-Route Gate-Level Simulation with Timing Use the following commands to perform post-map and place-and-route gate-level simulation with timing:

```
alib work
acom -work work <design_name>.vho <test_bench>.vhd
asim -t ps -noglitchmsg -sdfmax
/<instance_name>="<design_name>.sdf" work.<test_bench>
```

Note

The "/" character is Aldec's default hierarchy separator.

See Also "Third-Party Simulators" on page 315

Performing Simulation with Aldec Active-HDL

You can use the same procedure as in "Performing Simulation with Aldec Riviera Pro" on page 319 for creating or updating Lattice Semiconductor's FPGA (vendor) libraries and simulating Verilog or VHDL designs in Active-HDL.

Running Aldec Active-HDL in Stand-Alone Mode If you are running Active-HDL in stand-alone mode—that is, you are not accessing it through the Diamond graphical user interface—you should do the following to avoid elaboration (ELBREAD) errors when selecting top-level unit or compiling files from the design pop-up menu:

1. From the top-level menu, select **Design > Settings**.

2. Under Category in the left pane of the Design Settings dialog box, select **General > Compilation > Verilog**.
3. In the right (Verilog) pane under Verilog libraries, click the **Add Library** button  .
4. Select the reference Verilog libraries that are needed in your design. You can select multiple libraries by holding the CTRL key.
5. Click **OK**.
6. Click **OK** in the Design Settings dialog box.

This design setting also adds the -l <*verilog_library_name*> option to the vlog command, then it recognizes (or allows you to set) the top-level unit.

After setting the top-level unit (or testbench), you can start the simulation by clicking **Simulation > Initialize Simulation** from the top-level menu.

Note

If you are compiling and simulating your design by using a .do file, the steps just given are not needed. Instead, you should add -l with vlog, add -L with vsim, or both in your .do file.

If you are running Active-HDL in stand-alone mode and you are using at least one of the ASIC blocks, such as PCS or SYSBUS, in your design, you must do the following:

1. Change the directory to the design folder, that is, where the PCS or SYSBUS configuration file resides:

   ```
   cd <design_folder>
   ```

2. Set the simulation directory to the design (current) folder:

   ```
   set SIM_WORKING_FOLDER .
   ```

Setting the SIM_WORKING_FOLDER option forces the Active-HDL simulator to look for the configuration or input files in the design folder (by default Active-HDL looks for input files in the <*work*> and <*work*>/src folders only, where <*work*> is the Active-HDL working library folder). If you do not set this option, the simulation will fail and Active-HDL will issue an error about a missing configuration file.

This option is also necessary if your simulation is expecting any input files in the design folder rather than in the <*work*> or <*work*>/src folders, unless you place the input file path in your HDL code.

See Also "Third-Party Simulators" on page 315

Performing Simulation with Cadence NC-Verilog

This section explains how to perform simulation with Cadence NC-Verilog.

Setting Lattice Semiconductor Libraries Before simulating Lattice Semiconductor FPGA designs in the Cadence NC-Verilog simulator, you must perform the following steps to specify the Lattice Semiconductor simulation libraries.

Creating Library Definition Files Before using the NC-Verilog simulator to simulate your design project, you must first create two library definition files named hdl.var and cds.lib in your project folder. The hdl.var and cds.lib files define which libraries are accessible and where they are located. The hdl.var file contains optional statements that can be used to define variables such as the default work library, and the cds.lib file contains statements that map logical library names to their physical directory paths.

For example, your local hdl.var file may include the following:

```
DEFINE work work
```

Your local cds.lib file can include the following:

```
DEFINE ecp5u_vlog <compile_dir>/ecp5u_vlog
DEFINE ecp5um_vlog <compile_dir>/ecp5u_vlog
DEFINE ec_vlog <compile_dir>/ec_vlog
DEFINE ecp_vlog <compile_dir>/ecp_vlog
DEFINE ecp2_vlog <compile_dir>/ecp2_vlog
DEFINE ecp2m_vlog <compile_dir>/ecp2_vlog
DEFINE ecp3_vlog <compile_dir>/ecp3_vlog
DEFINE xp_vlog <compile_dir>/xp_vlog
DEFINE xp2_vlog <compile_dir>/xp2_vlog
DEFINE machxo_vlog <compile_dir>/machxo_vlog
DEFINE machxo2_vlog <compile_dir>/machxo2_vlog
DEFINE machxo3d_vlog <compile_dir>/machxo3d_vlog
DEFINE machxo3l_vlog <compile_dir>/machxo3l_vlog
DEFINE lptm_vlog <compile_dir>/lptm_vlog
DEFINE lptm2_vlog <compile_dir>/lptm2_vlog
DEFINE lifmd_vlog <compile_dir>/lifmd_vlog
DEFINE lifmdf_vlog <compile_dir>/lifmdf_vlog
DEFINE lfmnx_vlog <compile_dir>/lfmnx_vlog
DEFINE sc_vlog <compile_dir>/sc_vlog
DEFINE scm_vlog <compile_dir>/sc_vlog
DEFINE work ./work
```

Note

<compile_dir> refers to the location of the compiled device libraries.

Cadence provides a utility called NCLaunch to set up the necessary initialization files and to compile the Verilog source libraries. NCLaunch is available as part of the 2.1 and later releases. Otherwise, setting up the initialization files and compiling the Verilog source libraries is a manual process.

You can create the hdl.var and cds.lib files with any text editor. You must map the physical locations to the logical names before proceeding to the next step. On the Linux platform, you can create these names by using the mkdir command as follows:

```
mkdir -p <compile_dir>/ecp5u_vlog
```

```
mkdir -p <compile_dir>/ec_vlog
mkdir -p <compile_dir>/ecp_vlog
mkdir -p <compile_dir>/ecp2_vlog
mkdir -p <compile_dir>/ecp3_vlog
mkdir -p <compile_dir>/xp_vlog
mkdir -p <compile_dir>/xp2_vlog
mkdir -p <compile_dir>/machxo_vlog
mkdir -p <compile_dir>/machxo2_vlog
mkdir -p <compile_dir>/machxo3d_vlog
mkdir -p <compile_dir>/machxo3l_vlog
mkdir -p <compile_dir>/lptm_vlog
mkdir -p <compile_dir>/lptm2_vlog
mkdir -p <compile_dir>/lifmd_vlog
mkdir -p <compile_dir>/lifmdf_vlog
mkdir -p <compile_dir>/lfmnx_vlog
mkdir -p <compile_dir>/sc_vlog
```

If you want the logical library names to be available for all designs, use INCLUDE or SOFTINCLUDE in the location of your master hdl.var and cds.lib files.

For example, you can add the following line to your master cds.lib file:

```
INCLUDE $path/cds.lib
```

Note

The *path* variable specifies the location of your own cds.lib file.

Or, you can directly include the library definitions in the master hdl.var and cds.lib files.

Now the master hdl.var and cds.lib files include the Lattice Semiconductor library definitions. The next time that you want to simulate Lattice Semiconductor designs in the NC-Verilog simulator, you do not need to create you own hdl.var and cds.lib files again.

Parsing and Analyzing Lattice Semiconductor Simulation Libraries

After creating your own hdl.var and cds.lib files, you must parse and analyze the Lattice Semiconductor simulation libraries by using the NC-Verilog simulator. These libraries must be parsed and analyzed only once.

To parse and analyze Lattice Semiconductor Verilog simulation libraries, run the following commands in the NC-Verilog simulator:

- ECP5U Verilog:

  ```
  ncvlog -messages -work ecp5u_vlog
  $Diamond_install_path/cae_library/simulation/verilog/ecp5u/
  *.v
  ```

- LatticeEC Verilog:

  ```
  ncvlog -messages -work ec_vlog
  $Diamond_install_path/cae_library/simulation/verilog/ec/*.v
  ```

- LatticeECP Verilog:

  ```
  ncvlog -messages -work ecp_vlog
  ```

```
$Diamond_install_path/cae_library/simulation/verilog/ecp/*.v
```

- LatticeECP2 Verilog:

```
ncvlog -messages -work ecp2_vlog
$Diamond_install_path/cae_library/simulation/verilog/ecp2/
*.v
```

- LatticeECP3 Verilog:

```
ncvlog -messages -work ecp3_vlog
$Diamond_install_path/cae_library/simulation/verilog/ecp3/
*.v
```

- LatticeXP Verilog:

```
ncvlog -messages -work xp_vlog
$Diamond_install_path/cae_library/simulation/verilog/xp/*.v
```

- LatticeXP2 Verilog:

```
ncvlog -messages -work xp2_vlog
$Diamond_install_path/cae_library/simulation/verilog/xp2/*.v
```

- MachXO Verilog:

```
ncvlog -messages -work machxo_vlog
$Diamond_install_path/cae_library/simulation/verilog/machxo/
*.v
```

- MachXO2 Verilog:

```
ncvlog -messages -work machxo2_vlog
$Diamond_install_path/cae_library/simulation/verilog/
machxo2/*.v
```

- MachXO3D Verilog:

```
ncvlog -messages -work machxo3d_vlog
$Diamond_install_path/cae_library/simulation/verilog/
machxo3d/*.v
```

- MachXO3L Verilog:

```
ncvlog -messages -work machxo3l_vlog
$Diamond_install_path/cae_library/simulation/verilog/
machxo3l/*.v
```

- Platform Manager Verilog:

```
ncvlog -messages -work lptm_vlog
$Diamond_install_path/cae_library/simulation/verilog/lptm/
*.v
```

- Platform Manager 2 Verilog:

```
ncvlog -messages -work lptm2_vlog
$Diamond_install_path/cae_library/simulation/verilog/lptm2/
*.v
```

- LatticeSC Verilog:

```
ncvlog -messages -work sc_vlog
$Diamond_install_path/cae_library/simulation/verilog/sc/*.v
```

- LIFMD Verilog:

```
ncvlog -messages -work lifmd_vlog
$Diamond_install_path/cae_library/simulation/verilog/lifmd/
*.v
```

- LIFMDF Verilog:

```
ncvlog -messages -work lifmdf_vlog
$Diamond_install_path/cae_library/simulation/verilog/lifmdf/
*.v
```

- LFMNX Verilog:

```
ncvlog -messages -work lfmnx_vlog
$Diamond_install_path/cae_library/simulation/verilog/lfmnx/
*.v
```

Functional RTL Simulation Use the following commands to perform a functional RTL simulation with one of the libraries shown in Table 16:

```
rm –rf work
mkdir work
ncvlog -work work <design_name>.v <test_bench>.v
ncelab -lib_binding –access +rwc work.<test_bench>
ncsim -GUI work.<test_bench>
```

Note

If you are using NCSim version 5.4 or older, remove the –lib_binding option from the ncelab command.

Post-Map and Place-and-Route Gate-Level Simulation Use the following commands to perform post-map and place-and-route gate-level simulation:

```
rm –rf work
mkdir work
ncvlog -work work <design_name>.vo <test_bench>.v
ncelab -lib_binding –access +rwc work.<test_bench>
ncsim -GUI work.<test_bench>
```

Post-Map and Place-and-Route Gate-Level Simulation with Timing Use the following procedure to perform post-map and place-and-route gate-level simulation with timing:

1. Add the sdf_annotate line at the top-level test bench:

```
$sdf_annotate ("<design_name>.sdf", <design_name>,
,<sdf_log>, "MAXIMUM");
```

2. Use the following commands to perform the simulation:

```
rm –rf work
mkdir work
ncvlog -work work <design_name>.vo <test_bench>.v
ncelab -lib_binding –access +rwc work.<test_bench>
ncsim -GUI work.<test_bench>
```

See Also “Third-Party Simulators” on page 315

Performing Simulation with Cadence NC-VHDL

This section explains how to perform simulation with Cadence NC-VHDL.

Setting Lattice Semiconductor Libraries Before simulating Lattice Semiconductor FPGA designs in the Cadence NC-VHDL simulator, you must perform the following steps to set the Lattice Semiconductor simulation libraries.

Creating Library Definition Files Before using the Cadence NC-VHDL simulator to simulate your design project, you must first create two library definition files named hdl.var and cds.lib in your project folder. The hdl.var and cds.lib files define which libraries are accessible and where they are located. The hdl.var file contains optional statements that can be used to define variables such as the default work library, and the cds.lib file contains statements that map logical library names to their physical directory paths.

For example, your local hdl.var file can include the following:

```
DEFINE work work
```

Your local cds.lib file may include the following:

```
DEFINE ecp5u <compile_dir>/ecp5u_vhdl
DEFINE ecp5um <compile_dir>/ecp5u_vhdl
DEFINE ec <compile_dir>/ec_vhdl
DEFINE ecp <compile_dir>/ecp_vhdl
DEFINE ecp2 <compile_dir>/ecp2_vhdl
DEFINE ecp2m <compile_dir>/ecp2_vhdl
DEFINE ecp3 <compile_dir>/ecp3_vhdl
DEFINE xp <compile_dir>/xp_vhdl
DEFINE xp2 <compile_dir>/xp2_vhdl
DEFINE machxo <compile_dir>/machxo_vhdl
DEFINE machxo2 <compile_dir>/machxo2_vhdl
DEFINE machxo3d <compile_dir>/machxo3d_vhdl
DEFINE machxo3l <compile_dir>/machxo3l_vhdl
DEFINE lptm <compile_dir>/lptm_vhdl
DEFINE lptm2 <compile_dir>/lptm2_vhdl
DEFINE lifmd <compile_dir>/lifmd_vhdl
DEFINE lifmdf <compile_dir>/lifmdf_vhdl
DEFINE lfmnx <compile_dir>/lfmnx_vhdl
DEFINE sc <compile_dir>/sc_vhdl
DEFINE scm <compile_dir>/sc_vhdl
DEFINE work ./work
```

Note

<*compile*_dir> refers to the location of the compiled device libraries.

Cadence provides a utility called NCLaunch to set up the necessary initialization files and to compile the VHDL source libraries. NCLaunch is available as part of the 2.1 and later releases. Otherwise, setting up the initialization files and compiling the VHDL source libraries is a manual process.

You can create the hdl.var and cds.lib files with any text editor. You must map the physical locations to the logical names before you proceed to the next step. On the Linux platform, you can create these names by using the mkdir command as follows:

```
mkdir -p <compile_dir>/ecp5u_vhdl
mkdir -p <compile_dir>/ec_vhdl
mkdir -p <compile_dir>/ecp_vhdl
mkdir -p <compile_dir>/ecp2_vhdl
mkdir -p <compile_dir>/ecp3_vhdl
mkdir -p <compile_dir>/xp_vhdl
mkdir -p <compile_dir>/xp2_vhdl
mkdir -p <compile_dir>/machxo_vhdl
mkdir -p <compile_dir>/machxo2_vhdl
mkdir -p <compile_dir>/machxo3d_vhdl
mkdir -p <compile_dir>/machxo3l_vhdl
mkdir -p <compile_dir>/lptm_vhdl
mkdir -p <compile_dir>/lptm2_vhdl
mkdir -p <compile_dir>/lifmd_vhdl
mkdir -p <compile_dir>/lifmdf_vhdl
mkdir -p <compile_dir>/lfmnx_vhdl
mkdir -p <compile_dir>/sc_vhdl
```

If you want the logical library names to be available for all designs, use INCLUDE or SOFTINCLUDE in the location of your master hdl.var and cds.lib files.

For example, you can add the following line to your master cds.lib file:

```
INCLUDE $path/cds.lib
```

Note

The *path* variable specifies the location of your own cds.lib file.

Or, you can directly include the library definitions in the master hdl.var and cds.lib files.

Now the master hdl.var and cds.lib files include the Lattice Semiconductor library definitions. The next time that you want to simulate Lattice Semiconductor designs in the NC-VHDL simulator, you do not need to create you own hdl.var and cds.lib files again.

In addition to defining Lattice Semiconductor VHDL libraries, you must map the vital2000 logical library to the IEEE library location by adding the following line to your cds.lib file:

```
DEFINE vital2000 <NC_VHDL_Libraries_Folder>/IEEE
```

Note

The <*NC_VHDL_libraries_folder*> variable specifies the folder containing the NC-VHDL libraries.

Parsing and Analyzing Lattice Semiconductor Simulation Libraries
After creating your own hdl.var and cds.lib files, you must parse and analyze the Lattice Semiconductor simulation libraries by using the NC-VHDL simulator. These libraries must be parsed and analyzed only once.

To parse and analyze Lattice Semiconductor VHDL simulation libraries, run the following commands in NC-VHDL simulator.

- ECP5U VHDL:

```
ncvhdl -messages -work ecp5u -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/ecp5u/src/
*.vhd

ncvlog -messages -work ecp5u $Diamond_install_path/
cae_library/simulation/vhdl/ecp5u/src/*.v
```

- LatticeEC VHDL:

```
ncvhdl -messages -work ec -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/ec/src/
*.vhd
```

- LatticeECP VHDL:

```
ncvhdl -messages -work ecp -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/ecp/src/
*.vhd
```

- LatticeECP2 VHDL:

```
ncvhdl -messages -work ecp2 -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/ecp2/src/
*.vhd
```

- LatticeECP3 VHDL:

```
ncvhdl -messages -work ecp3 -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/ecp3/src/
*.vhd
```

- LatticeXP VHDL:

```
ncvhdl -messages -work xp -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/xp/src/
*.vhd
```

- LatticeXP2 VHDL:

```
ncvhdl -messages -work xp2 -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/xp2/src/
*.vhd
```

- LIFMD VHDL:

```
ncvhdl -messages -work lifmd-smartorder
$Diamond_install_path/cae_library/simulation/vhdl/lifmd/src/
*.vhd
```

- LIFMDF VHDL:

```
ncvhdl -messages -work lifmdf-smartorder
$Diamond_install_path/cae_library/simulation/vhdl/lifmdf/
src/*.vhd
```

- LFMNX VHDL:

```
ncvhdl -messages -work lfmnx-smartorder
$Diamond_install_path/cae_library/simulation/vhdl/lfmnx/src/
*.vhd
```

- MachXO VHDL:

```
ncvhdl -messages -work machxo -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/machxo/
src/*.vhd
```

- MachXO2 VHDL:

```
ncvhdl -messages -work machxo2 -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/machxo2/
src/*.vhd
```

- MachXO3D VHDL:

```
ncvhdl -messages -work machxo3d -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/machxo3d/
src/*.vhd
```

- MachXO3H VHDL:

```
ncvhdl -messages -work machxo3h -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/machxo3h/
src/*.vhd
```

- MachXO3L VHDL:

```
ncvhdl -messages -work machxo3l -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/machxo3l/
src/*.vhd
```

- Platform Manager VHDL:

```
ncvhdl -messages -work lptm -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/lptm/src/
*.vhd
```

- Platform Manager 2 VHDL:

```
ncvhdl -messages -work lptm2 -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/lptm2/src/
*.vhd
```

- LatticeSC VHDL:

```
ncvhdl -messages -work sc -smartorder
$Diamond_install_path/cae_library/simulation/vhdl/sc/src/
*.vhd
```

Functional RTL Simulation Use the following commands to perform a functional RTL simulation with one of the libraries shown in Table 2:

```
rm –rf work
mkdir work
ncvhdl -work work <design_name>.vhd <test_bench>.vhd
ncelab -lib_binding –access +rwc work.<test_bench>
ncsim –GUI work.<test_bench>
```

Note

If you are using NCSim version 5.4 or older, remove the –lib_binding option from the ncelab command.

If you receive warning messages during compilation, see "Warning Messages Caused by Std_Logic Declarations" on page 333.

Post-Map and Place-and-Route Gate-Level Simulation Use the following commands to perform post-map and place-and-route gate-level simulation:

```
rm –rf work
mkdir work
ncvhdl -work work <design_name>.vho <test_bench>.vhd
ncelab -lib_binding –access +rwc work.<test_bench>
ncsim –GUI work.<test_bench>
```

If you receive warning messages during compilation, see "Warning Messages Caused by Std_Logic Declarations" on page 333.

Post-Map and Place-and-Route Gate-Level Simulation with Timing Use the following procedure to perform post-map and place-and-route gate-level simulation with timing:

1. Compile the SDF file:

   ```
   ncsdfc <design_name>.sdf
   ```

   This command generates the compiled SDF file, <*design_name*>.sdf.X.

2. Create an <*sdf_cmd*> file with the following contents:

   ```
   COMPILED_SDF_FILE = "<design_name>.sdf.X", SCOPE =
   :<instance_name>,
   LOG_FILE = "<sdf_log>", MTM_CONTROL = "MAXIMUM";
   ```

3. Use the following commands to perform the simulation:

   ```
   rm –rf work
   mkdir work
   ncvhdl -work work <design_name>.vho <test_bench>.vhd
   ncelab -lib_binding –access +rwc –sdf_cmd_file <sdf_cmd>
   work.<test_bench>
   ncsim –GUI work.<test_bench>
   ```

If you receive warning messages during compilation, see "Warning Messages Caused by Std_Logic Declarations" on page 333.

Warning Messages Caused by Std_Logic Declarations When you set vital_level1 as an attribute for the architecture, Cadence NC-VHDL issues several warning messages during compilation about an unapproved type in the vital2000 library models. These messages are caused by Cadence NC-VHDL's strict interpretion of std_logic signal declarations in the architecture as signal types. You can make this interpretation less strict by using the following option:

```
ncvhdl -relax
```

This option does not affect the actual simulation.

See Also "Third-Party Simulators" on page 315

Performing Mixed-Language (Verilog and VHDL) Simulation with Cadence NCSim

The steps involved in performing Verilog/VHDL mixed-language simulation with Cadence NCSim are straightforward and use most of the information provided in the previous two sections ("Simulation Library Files" on page 317 and "Performing Simulation with Cadence NC-VHDL" on page 329):

1. Set up your local cds.lib and hdl.var files to include all the libraries that you use in your design. Each library can be Verilog only, VHDL only, or a mixed-language library containing both VHDL entities and Verilog modules. The libraries can be Lattice Semiconductor FPGA libraries, protected libraries (for example, special-block or IP libraries), or your own design libraries.
2. Compile your Verilog files, using the ncvlog command.
3. Compile your VHDL files, using the ncvhdl command.
4. Elaborate the top-level unit (or test bench) in your mixed-language design, using the ncelab command.
5. Simulate the top-level unit (or test bench), using the ncsim command.

Following is an example of the procedure used for simulating a mixed-language design that consists of your own VHDL design files, the LatticeSC VHDL source files, and the Lattice PCSA encrypted Verilog module:

1. Compile the LatticeSC VHDL source files into the LatticeSC VHLD library, as explained in "Performing Simulation with Cadence NC-VHDL" on page 329.
2. Compile the Lattice PCSA encrypted Verilog source file into the same LatticeSC VHDL library as follows:

   ```
   ncvlog -messages -work sc <install_dir>/cae_library/
   simulation/blackbox/pcsa-ncv.vp
   ```

3. Create a local folder for the work library:

   ```
   mkdir work
   ```

4. Add the following lines to your local hdl.var file:

   ```
   DEFINE WORK work
   ```

5. Add the following lines to your local cds.lib:

   ```
   DEFINE sc <Path to NCSim-compiled SC VHDL lib>
   DEFINE work ./work
   ```

6. Compile your VHDL design and test bench files:

   ```
   ncvhdl <design_name>.vhd <test_bench>.vhd
   ```

7. Elaborate your top-level unit (test bench):

   ```
   ncelab -lib_binding –NOMXINDR -access +rwc <test_bench>
   ```

   The –NOMXINDR option is added to ncelab to avoid MXINDR errors. This option is necessary for mixed-language designs.

8. Simulate your test bench:

```
ncsim -GUI <test_bench>
```

Performing Simulation with Mentor Graphics ModelSim / Questa

This section explains how to perform simulation with the Mentor Graphics ModelSim / Questa simulator.

Setting Lattice Semiconductor Libraries Before simulating Lattice Semiconductor FPGA designs in the Mentor Graphics ModelSim / Questa simulator, you must compile the Lattice Semiconductor simulation libraries using the cmpl_libs command available from the Diamond's TCL console.

For details about using this command, refer to “Simulation Libraries Compilation Tcl Command” on page 2606.

To enable Diamond’s batch interface to ModelSim / Questa, change the default simulation tool setup and directory reference as follows:

1. Choose **Options > Environment Options**.
2. Click on the **Directories** tab.
3. Change the path setting to the ModelSim / Questa executable on the tab to the new version.
4. Click **OK**.

Mapping to ModelSim / Questa Libraries The library compilation target folder should also be your simulation working directory. Otherwise, you need to copy the library mappings from modelsim.ini at the target folder (or the file itself) into your local or master modelsim.ini.

After running simulation libraries compilation using the cmpl_libs TCL command, a modelsim.ini file will be created at the target folder chosen for the compiled libraries. That file will have the proper mappings for these compiled libraries.

Performing Verilog Simulation Use the procedures described in this section to perform a Verilog standalone simulation with ModelSim / Questa.

Functional RTL Simulation Use the following commands to perform a functional RTL simulation with one of the libraries shown in Table 16:

```
vlib work
vlog -work work <design_name>.v <test_bench>.v
```

```
vsim work.<test_bench>
```

Note

If you have a pre-existing work library, you can clear its contents by using the following command:

```
vdel -lib work -all
```

Post-Map and Place-and-Route Gate-Level Simulation Use the following commands to perform post-map and place-and-route gate-level simulation:

```
vlib work
vlog -work work <design_name>.vo <test_bench>.v
vsim -L <library_name> work.<test_bench>
```

where <*library_name*> is the FPGA Verilog library to be searched for the low-level units.

Post-Map and Place-and-Route Gate-Level Simulation with Timing Use the following procedure to perform post-map and place-and-route gate-level simulation with timing:

1. Add the `sdf_annotate` line at the top-level test bench:

   ```
   $sdf_annotate ("<design_name>.sdf", <instance_name>,
   ,<sdf_log>, "MAXIMUM");
   ```

2. Use the following commands to perform the simulation:

   ```
   vlib work
   vlog -work work <design_name>.vo <test_bench>.v
   vsim -L <library_name> work.<test_bench>
   ```

Performing VHDL Simulation Use the procedures described in this section to perform a VHDL standalone simulation with ModelSim / Questa.

Functional RTL Simulation Use the following commands to perform a functional RTL simulation with one of the libraries shown in Table 17:

```
vlib work
vcom -work work <design_name>.vhd <test_bench>.vhd
vsim -t ps work.<test_bench>
```

Note

If you have a pre-existing work library, you can clear its contents by using the following command:

```
vdel -lib work -all
```

Post-Map and Place-and-Route Gate-Level Simulation Use the following commands to perform post-map and place-and-route gate-level simulation:

```
vlib work
vcom -work work <design_name>.vho <test_bench>.vhd
vsim -t ps +no_glitch_msg work.<test_bench>
```

Post-Map and Place-and-Route Gate-Level Simulation with Timing Use the following commands to perform post-map and place-and-route gate-level simulation with timing:

```
vlib work
vcom -work work <design_name>.vho <test_bench>.vhd
vsim -t ps +no_glitch_msg -sdfmax
<instance_name>="<design_name>.sdf" work.<test_bench>
```

See Also "Third-Party Simulators" on page 315

Performing Simulation with Synopsys VCS

This section explains how to perform simulation with Synopsys VCS for Verilog-only designs.

For VHDL designs, refer to "Performing Mixed-Language (Verilog and VHDL) Simulation with Synopsys VCS-MX" on page 338.

For SystemVerilog designs, refer to "Performing Mixed-Language (Verilog and SystemVerilog) Simulation with Synopsys VCS-MX" on page 339.

Functional RTL Simulation Use the following vcs command to perform a functional RTL simulation with one of the libraries shown in Table 16:

```
vcs -RI <design_name>.v <test_bench>.v
-y <Lattice_verilog_library_location> +libext+.v +v2k
```

Post-Map and Place-and-Route Gate-Level Simulation Use the following vcs command to perform post-map and place-and-route gate-level simulation:

```
vcs -RI <design_name>.vo <test_bench>.vo
-y <Lattice_verilog_library_location> +libext+.v +v2k
```

Post-Map and Place-and-Route Gate-Level Simulation with Timing Use the following procedure to perform post-map and place-and-route gate-level simulation with timing:

1. Add the `sdf_annotate` line at the top-level test bench:

   ```
   $sdf_annotate ("<design_name>.sdf", <instance_name>,
   ,<sdf_log>, "MAXIMUM");
   ```

2. Use the following `vcs` command to perform the simulation:

   ```
   vcs -RI <design_name>.vo <test_bench>.v
   -y <Lattice_verilog_library_location> +libext+.v +v2k
   ```

See Also "Third-Party Simulators" on page 315

Performing Mixed-Language (Verilog and VHDL) Simulation with Synopsys VCS-MX

In order to run mixed-language simulation with VCS, you must have a special license for VCS-MX. Contact Synopsys if you do not have one.

"Performing Simulation with Synopsys VCS" on page 337 describes how to simulate a Verilog-only design by simply using the vcs command. In order to simulate a VHDL-only or a Verilog/VHDL mixed-language design, extra steps are required (with extra commands). In general, if your design contains VHDL source code, you must analyze it by using the vhdlan utility. And, if your VHDL design instantiates a Verilog design within it, you must use the vlogan utility. Mixed-language designs are categorized into either VHDL-top or Verilog-top designs, where each one has its own procedure for running the simulation with VCS-MX tools. For more details, refer to the *VCS MX/VCS MXi User Guide*.

Following is an example of the procedure used for simulating a Verilog/VHDL mixed-language design that consists of your own VHDL design files, the ECP5U VHDL library source files, and the ECP5U black boxes encrypted Verilog source file.

1. Create local folders for the ECP5U VHDL and the work libraries:

   ```
   mkdir ecp5u_vhdl work
   ```

2. Create (or modify) your local synopsys_sim.setup file, and add the following lines:

   ```
   WORK > DEFAULT
   DEFAULT : ./work
   ECP5U: ./ecp5u_vhdl
   ```

3. Analyze the ECP5U VHDL library:

   ```
   vhdlan -work ECP5U <ecp5u_vhdl_src_folder>/ECP5UCOMP.vhd \
   <ecp5u_vhdl_src_folder>/ECP5U_SEQ.vhd \
   <ecp5u_vhdl_src_folder>/ECP5U_IO.vhd \
   <ecp5u_vhdl_src_folder>/ORCA_CMB.vhd \
   <ecp5u_vhdl_src_folder>/ORCA_MEM.vhd \
   <ecp5u_vhdl_src_folder>/ORCA_MISC.vhd \
   <ecp5u_vhdl_src_folder>/ORCA_LUT.vhd \
   <ecp5u_vhdl_src_folder>/gsr_pur_assign.vhd
   vlogan -work ECP5U <ecp5u_vhdl_src_folder>/*.v
   vhdlan -work ECP5U <ecp5u_vhdl_src_folder>/ECP5U_SL.vhd
   ```

   where <*ecp5u_vhdl_src_folder*> refers to <*install_dir*>/cae_library/simulation/vhdl/ecp5u/src.

4. Analyze the Lattice black box modules from the encrypted Verilog source as follows:

   ```
   vlogan -work ECP5U <install_dir>/cae_library/simulation/
   blackbox/ecp5u_black_boxes-vcs.vp
   ```

5. Analyze your VHDL design and test bench files:

```
vhdlan <design_name>.vhd <test_bench>.vhd
```

6. Compile the simulation executable from your top-level unit (test bench):

```
scs -mhdl <test_bench>
```

7. Simulate your test bench by running the simulation executable:

```
scsim
```

See Also "Third-Party Simulators" on page 315

Performing Mixed-Language (Verilog and SystemVerilog) Simulation with Synopsys VCS-MX

In order to run mixed-language simulation with VCS, you must have a special license for VCS-MX. Contact Synopsys if you do not have one.

"Performing Simulation with Synopsys VCS" on page 337 describes how to simulate a Verilog-only design by simply using the vcs command. In order to simulate a Verilog/SystemVerilog mixed-language design, extra steps are required. In general, if your design contains SystemVerilog code, then you need first to compile the Lattice library sources and/or encrypted modules using the vlogan utility. For more details about running mixed-language designs, refer to the *VCS MX/VCS MXi User Guide*.

Following is an example of the procedure used for simulating a Verilog/ SystemVerilog mixed-language design that consists of your own SystemVerilog design files, modules from the MachXO2 Verilog source files and the machxo2_black_boxes encrypted Verilog source file.

1. Create local folder for the work library:

```
mkdir work
```

2. Create (or modify) your local synopsys_sim.setup file, and add the following lines:

```
WORK > DEFAULT
DEFAULT : ./work
```

3. Analyze the machxo2_black_boxes modules from their encrypted Verilog source file as follows:

```
vlogan +v2k <install_dir>/cae_library/simulation/blackbox/
machxo2_black_boxes-vcs.vp
```

4. Compile your SystemVerilog files and simulate your design with top-level unit (test bench):

```
vcs -mhdl -RI <design_name>.v <test_bench>.v -y
<install_dir>/cae_library/simulation/Verilog/machxo2
+libext+.v
```

See Also "Third-Party Simulators" on page 315

Referencing Black-Box Models

Some high-value models, such as ECP5UM DCU, are not distributed as HDL open source. In these cases, either HDL encrypted sources or pre-compiled libraries are provided for the respective simulator. By default, Diamond installs archives for each encrypted or pre-compiled model in the following directory:

`<install_dir>\cae_library\simulation\blackbox`

For older devices (LatticeSC, LatticeEC, LatticeECP, LatticeECP2, LatticeECP3, LatticeXP, LatticeXP2, Platform Manager and MachXO), each file or folder name indicates the model and simulator supported using the following format:

`<model>[-aldec|-ncv|-vcs].vp | folder <model>/`

where:

- <*model*> is the name of the Verilog simulation model.
- `-aldec` indicates that the model is provided as encrypted Verilog source for the Aldec Riviera Pro simulator. These models can also be compiled for Aldec Active-HDL simulator.
- `-ncv` indicates that the model is provided as encrypted Verilog source for the Cadence NCSim simulator.
- `-vcs` indicates that the model is provided as encrypted Verilog source for the Synopsys VCS simulator.
- <*model*>/ is a folder indicating that the model is available as a pre-compiled library for the Mentor Graphics ModelSim / Questa simulator, but it is recommended to use the Lattice-provided pre-compiled libraries for Modelsim simulator located at (*install_folder*)/modeltech/lib.

For example, the file JTAGA-vcs.vp contains the JTAGA encrypted Verilog model for Synopsys VCS.

For newer devices (ECP5U, MachXO2, MachXO3D, MachXO3L and Platform Manager 2), all the black-box models for each device family are contained in a single .vp file or a single folder for each simulator, in the following format:

`<device>_black_boxes[-aldec|-ncv|-vcs].vp | folder <device>_black_boxes/`

Some Verilog models are pre-compiled or encrypted differently for VHDL. Those models will have the following format:

`[<model>|<device>_black_boxes]_vhdlib[-aldec|-ncv|-vcs].vp | folder [<model>|<device>_black_boxes]_vhdlib/`

If the models are supplied as encrypted Verilog models, you must compile them with other source files for the same device into the same library (both Verilog and VHDL) for the respective simulator. Refer to each respective section for each third-party simulator in "Third-Party Simulators" on page 315.

For ModelSim / Questa pre-compiled libraries, if you have a higher version of ModelSim / Questa simulator, you need to refresh the library as follows:

```
vlog -refresh -work <library_name>
```

For VCS encrypted sources, make sure you compile the files for Verilog, but not for SystemVerilog. For example, use the +v2k option.

If a README file is provided, examine it in each archive file for any further compilation instructions.

See Also "Third-Party Simulators" on page 315

Using the Global Set/Reset (GSR) Signal

Lattice FPGA device simulation models provide the Global Set/Reset Interface (GSR) signal for use with all Lattice FPGAs.

Details regarding the architecture of global signals are described in the data sheet for each device family on the Lattice Web site.

For more information on the GSR library element, see "FPGA Libraries Reference Guide" on page 1665.

For more information on GSR and RTL Functional Simulation, see "RTL Functional Simulation and the GSR Resource" on page 1179.

For more information on how to use the GSR signal, see "How to Use the Global Set/Reset (GSR) Signal" on page 1174.

Creating Waveform Test Stimulus

Once you have completed your design (or a module of the design), you will want to test it to confirm that it behaves the way you expect it to. Simulation requires a test stimulus file that specifies the input waveforms.

You can graphically create a waveform test stimulus file with the Waveform Editor. You see exactly what each waveform will look like, as well as its timing relationship to all the other waveforms.

The Waveform Editor window consists of two panes. The left-hand Waveform Editing area is used for graphically creating and editing waveforms. The right-hand pane includes three views for setting pulse/waveform properties and using patterns.

The following design flow illustrates how you would typically create a stimulus file with the Waveform Editor.

1. Create a waveform file (.wdl).
2. Set Time Unit for the simulation.
3. Select the signals for which you want to define stimuli.
4. Draw the stimulus for each input.
5. Export the waveform information to a stimulus file.

Creating a Waveform File

You can create a new waveform file (.wdl) in Lattice Diamond.

To create a new .wdl file:

1. In the Diamond window, choose **File > New > File**.
2. In the New File dialog box, do the following:
   - Under Other Files, select **WaveForm Files**.
   - Specify the file name and location.
   - Select **Add to project** if you want to include the new file in the current project directory.
   - Select an implementation name from the Implementation name drop-down list.
   - Click **New**.

   The Waveform Editor opens with a blank window.

Setting Simulation Time Unit

You can select the Time Unit for the simulation database. In the simulation stimulus file, stimulus values are specified at intervals of the specified Time Unit.

All stimulus events occur at, and have lengths of, integral multiples of the Time Unit. This feature makes it easier to draw waveforms because all Transitions are automatically forced to occur at multiples of the Time Unit.

Note

A long Simulation Time combined with short Time Units may require an excessive amount of computer resources. Be sure you have selected duration values appropriate for your design before you begin simulation.

To set simulation time unit:

1. In the Waveform Editor, choose **Edit > Timing Settings**.

   The Time Settings dialog box opens.

2. Select the time units you want and click **OK**.

Creating Waveforms

Before drawing waveforms in the Waveform Editor, you may need to specify time unit for the simulation. See “Setting Simulation Time Unit” on page 342.

If you make an error when creating waveforms, choose **Edit > Undo**. The Waveform Editor allows you to perform an unlimited number of Undos and Redos until you save the file.

To create a waveform:

1. Be sure you are in the Edit mode. If not, choose **Edit > Edit Mode** or click the  icon on the toolbar to switch to the Edit mode.
2. At the bottom of the right-hand pane, click the **Pulse** tab to display the Pulse Properties view.

   Note

   If the Pulse tab is not available, choose **View > Pulse Properties View** to display it.

3. Choose **Edit > New Wave**.

   The Add New Wave dialog box opens.

4. In the dialog box, do the following:
   - Type a name for the new wave in the Wave Name box.
   - Select the polarity for the new waveform.
   - Click **Add**.

     The new waveform name is displayed in the Waveform Editing area. The letter behind the waveform name indicates its polarity. “I” stands for Input, “O” for Output, “B” for Bidirectional, “T” for Test Point, and “G” for Global. A pattern name begins with a dollar sign ($).

- You may keep the dialog box open and continue to add more waveform names. When you finish, click **Close**.

5. In the Waveform Editing area, click the new waveform name to highlight it.
6. Move the mouse to the waveform section of the window and click where you want the first Transition. (You must click inside the area next to the highlighted name.)

   A blue line is drawn showing the first section (called a Pulse) of the waveform. You can also see two black horizontal lines above and below the new pulse, indicating it is currently selected.
7. Keep the pulse selected. Then do the following in the Pulse Properties view.
   - In the State box, select a state for the selected pulse and press **Enter**.

     If you are adding a bus pulse, you can select **Set Value** in the State box, and then enter the value in the Value box below it. The value entered in the Value box must be in hexadecimal format.
   - In the Duration box, enter a time value.

     You don't have to enter the time unit. The Waveform Editor automatically appends the time unit specified in the Time Settings dialog box to the time value. To apply a different time unit, see "Setting Simulation Time Unit" on page 342.
8. In the Waveform Editing area, click again to create the second Transition.

   The pulse drawn has the opposite state of the first pulse.
9. Set the properties for the second pulse.
10. Continue clicking and setting properties to complete the waveform.
11. Choose **File > Save File** to save the new waveform.

Note

The commands on the View menu allow you to zoom in and zoom out on the waveforms, to view them in more or less detail.

Creating Bus Waveforms

You can create bus waveforms in the Waveform Editor. Bus waveforms can be expanded to separate waveforms in the Waveform Editing area.

Bus waveform pulses are displayed as elongated hexagons, as shown below.

Figure 48:



Tip

You can double-click the plus sign before a bus name to display the individual waveforms included in the bus. However, you cannot edit those individual waveforms. You can only edit the bus waveform.

To create a bus waveform:

1. Be sure you are in the Edit mode. If not, choose **Edit > Edit Mode** or click the icon on the toolbar to switch to the Edit mode.
2. At the bottom of the right-hand pane, click the **Pulse** tab to display the Pulse Properties view.

   Note

   If the Pulse tab is not available, choose **View > Pulse Properties View** to display it.

3. Choose **Edit > New Wave**.

   The Add New Wave dialog box opens.
4. In the Wave Name box, enter the bus name by adding a bit range to the name. See the following examples:

   A[0:2], which you can expand to A[0], A[1], and A[2].

   A, B, C, D, which you can expand to A, B, C, and D.

   [A1...A3], which you can expand to A1, A2, and A3.
5. Select Polarity for the new bus waveform.
6. Click **Add** and then **Close**.

   The new bus name is displayed in the Waveform Editing area. You can double-click the plus sign before the bus name to expand the bus.
7. Click the new bus name to highlight it.
8. Move the mouse to the waveform section of the window and click where you want the first Transition. (You must click inside the area next to the highlighted name.)

   A bus pulse is drawn showing the first section of the bus waveform. You can also see two black horizontal lines above and below the bus pulse, indicating it is currently selected.
9. Keep the pulse selected. Then do the following in the Pulse Properties view.

   - In the State box, select **Don't Care** or **High-Z**, or select **Set Value** and then enter the bus value in the Value box below it. The value entered in the Value box must be in hexadecimal format.

     Bus pulses can have any value that can be represented by the number of bits in the bus. For example, any pulse in an eight-bit bus waveform can be assigned a value between 0 and 255. Bus pulses are given default values, in sequential order: 0, 1, 2, 3, and so on. You can change any of these values at any time.

The format of the bus value displayed in the Waveform Editing area is set by the Bus Radix option in the Options dialog box. To select a different Bus Radix format, see “Changing the Waveform Display Options” on page 358.

- In the Duration box, enter a time value.

  You don’t have to enter the time unit. The Waveform Editor automatically appends the time unit specified in the Time Settings dialog box to the time value. To apply a different time unit, see “Setting Simulation Time Unit” on page 342.

10. In the Waveform Editing area, click again to create the second Transition, and set the properties for the second bus pulse.
11. Continue clicking and setting properties to complete the bus waveform.
12. Choose **File > Save File** to save the new bus waveform.

Note

The commands on the View menu allow you to zoom in and zoom out on the waveforms, to view them in more or less detail.

Creating and Using Patterns

You may find yourself drawing the same waveform over and over again. The Waveform Editor allows you to name and define arbitrary waveforms, called Patterns, which you can then add to or insert in any other waveform. After a pattern is inserted to another waveform, it becomes a series of pulses, and loses its identity as a pattern. When you modify a pattern, the waveforms that use that pattern will not change.

A pattern name begins with a dollar sign ($). Patterns normally represent single-bit data. You can create a bus pattern by adding a bit range to the name, as shown below:

$buspat[7:0]

Bus patterns cannot be added to single-bit waveforms, or vice versa. Also, the number of bits in a bus pattern must match the number of bits in the bus waveform to which it is added.

To add a pattern name in the Waveform Editor:

1. In the Waveform Editor, choose **Edit > New Wave**.

   The Add New Wave dialog box opens.
2. In the Polarity drop-down box, select **Pattern**.
3. In the Wave Name box, type a name for the pattern. The name should begin with a dollar sign ($). If you do not include a dollar sign, the Waveform Editor automatically adds one.

4. Click **Add** to add the pattern's name to the Waveform Editing area. The dialog box remains open, so you can continue to add pattern names without having to reselect the command.
5. Click **Close**.

You can then draw the waveform for your pattern.

To draw a pattern:

1. Be sure you are in the Edit mode. If not, choose **Edit > Edit Mode** or click the icon on the toolbar to switch to the Edit mode.
2. At the bottom of the right-hand pane, click the **Pulse** tab to display the Pulse Properties view.

   Note

   If the Pulse tab is not available, choose **View > Pulse Properties View** to display it.

3. In the Waveform Editing area, highlight the pattern name for which you want to create a waveform.
4. Move the mouse to the waveform section and click where you want the first Transition. (You must click inside the area next to the highlighted name.)

   A blue line is drawn showing the first section (called a Pulse) of the waveform. You can also see two black horizontal lines above and below the new pulse, indicating it is currently selected.
5. Keep the pulse selected. Then do the following in the Pulse Properties view.
   - In the State box, select a state for the selected pulse and press **Enter**.

     If you are adding a bus pulse, you can select **Set Value** in the State box, and then enter the value in the Value box below it. The value entered in the Value box must be in hexadecimal format.
   - In the Duration box, enter a time value.

     You don't have to enter the time unit. The Waveform Editor automatically appends the time unit specified in the Time Settings dialog box to the time value. To apply a different time unit, see "Setting Simulation Time Unit" on page 342.
6. In the Waveform Editing area, click again to create the second Transition.

   The pulse drawn has the opposite state of the first pulse.
7. Set the properties fro the second pulse.
8. Continue clicking and setting properties to complete the waveform.
9. Choose **File > Save File** to save the new pattern.

Once the pattern is drawn, you can add it to a waveform or another pattern.

To add a pattern to a waveform or another pattern:

1. Be sure you are in the Edit mode. If not, choose **Edit > Edit Mode** or click the  icon on the toolbar to switch to the Edit mode.
2. At the bottom of the right-hand pane, click the **Pattern** tab to display the Pattern view.

   The Pattern List box lists all the patterns defined in the current waveform file in alphabetical order.

   Note

   If the Pattern tab is not available, choose **View > Pattern View** to display it.

3. In the Waveform Editing area, select the pulse where you want to insert the pattern.
4. In the Pattern view, select the name of the pattern you want to add and click **Insert**.

   The pattern is inserted before the selected pulse.

You can also use the Copy command on the Edit menu to copy a pattern to the clipboard, and then use the Paste command to insert it in another waveform. The pattern, however, is inserted only as a series of pulses, and loses its identity as a pattern.

Importing Signal Names from a NAF File

A NAF file is an intermediate file generated by the Generate Schematic Symbol command in the Hierarchy view. It is a text file that lists signal names and types in design modules. A typical NAF file looks like the following:

```
c[2]  o
c[1]  o
c[0]  o
clk  i
rst  i
ExtTrigOutPort0  o
```

Importing signal names from a NAF file is much quicker than adding waveform names one by one using the Edit > New Wave command.

Note

For detailed information on how to run **Generate Schematic Symbol** in the Hierarchy view, see “Creating a Symbol from an HDL Module” on page 277.

To import signal names from a NAF file:

1. In the Waveform Editor, select **Edit > Import Wave**.

   The Import Wave dialog box opens.

2. Click **Browse** to display the Open Existing Waveform File dialog box.
3. Select a .naf file and click **Open**.

   All the signals contained in the .naf file are then listed in the Imported Wave box. You can narrow the list by using the Filter options on the left.

   Next you will add signal names to the Selected Wave box. Only the names listed in the Selected Wave box will be added to the waveform display.
4. In the Imported Wave box, click to highlight the signal names that you want to import. You can use the Shift or Ctrl key to highlight multiple names.
5. Click **>** to add the highlighted signal names to the Selected Wave box.

   Tip

   You can use the four arrow buttons in the middle to add or remove signal names in the Selected Wave box.

   > – Adds the signal names highlighted in the Imported Wave box to the Selected Wave box.

   >> – Adds all the signal names in the Imported Wave box to the Selected Wave box.

   < – Deletes the signal names highlighted in the Selected Wave box.

   << – Clears all signal names from the Selected Wave box.
6. Click **OK**. The signal names in the Selected Wave box are added to the waveform display.

After adding signal names, you can then start drawing waveforms for them. See “Creating Waveforms” on page 343 for detailed information on how to draw waveforms.

Adding Test Points

You can add waveforms for internal nodes (that is, nets that do not have external connections or I/O markers) in the Waveform Editor window. These internal nodes are called test points.

Test points can be used to “force” internal nodes to specific values during simulation. However, not all simulators support this function. Check your simulator documentation to see if you can define arbitrary values for internal nodes.

To add a test points from the Waveform Editor:

1. In the Waveform Editor, choose **Edit > New Wave**.

   The Add New Wave dialog box opens.
2. In the Wave Name box, type the name of the test point.
3. In the Polarity box, select **Test Pt**.
4. Click **Add** to add the test point to the Waveform Editing area.

After adding the Test Points, you can then start drawing waveforms for them. See “Creating Waveforms” on page 343 for detailed information on how to draw waveforms.

Selecting Waveforms or Pulses

You can select waveforms or pulses (a Pulse is a section of a waveform between two Transitions) in the Edit operation mode. If you are not in the Edit mode, choose **Edit > Edit Mode** or click the icon on the toolbar to switch to the Edit mode.

To select a waveform:

- Click the name of the waveform you want to select.

To select pulses:

- To select a single pulse, click anywhere within a pulse.
- To select multiple successive pulses, drag the cursor across the waveform to cover several pulses.

  The selected pulse(s) will be marked with thick black lines above and below it.

Graphically Changing the Length of a Pulse

In the Waveform Editor, a section of a waveform between two Transitions is called a Pulse. You can graphically change the length of any pulse. The length of a pulse can also be changed by specifying the pulse duration in the Pulse Properties view. See “Setting Pulse State and Duration” on page 351 for details.

To change the length of a pulse graphically:

1. Click the pulse you want to lengthen or shorten.

   The pulse becomes selected, marked with thick black lines above and below it.

2. Click and hold down the mouse button on the selected pulse.

   The selected pulse and its preceding or succeeding pulse (depending on where you click) both change to dotted lines.

3. Drag the mouse to the left or right to change the length of the pulse.
   - If you select the last pulse in a waveform, dragging changes the length of that pulse and extends or shortens the waveform.
   - If you select a pulse within the waveform, dragging alters the start or ending time of the Transition. The preceding or succeeding pulse is lengthened and the selected pulse shortened (or vice versa). The total

length of both pulses (and of the complete waveform) does not change.

- If you press and hold **Shift** before you click a pulse, dragging only changes the length of the selected pulse. The preceding or succeeding pulse lengths do not change. Any pulses to the left or right of the selected pulse are moved (without modification) to accommodate the lengthened or shortened pulse. Use this mode to adjust a pulse without changing anything else in the waveform.

Remember that pulses always have a length that is an integral multiple of the Time Units selected in the Timing Settings dialog box.

Setting Pulse Properties

In the Waveform Editor, a section of a waveform between two Transitions is called a Pulse. You can use the Pulse Properties view to change pulse properties, including state, duration, scaling, and repeat options.

To display the Pulse Properties view, click the **Pulse** tab at the bottom of the right-hand pane. If the Pulse tab is not displayed, choose **View > Pulse Properties View**.

Note

You cannot select a pulse or change pulse properties for individual waveforms included in a bus. However, you can select pulses on a bus waveform and change pulse properties.

Setting Pulse State and Duration

Before setting pulse state and duration, make sure you are in the Edit operation mode. If not, choose **Edit > Edit Mode** or click the icon on the toolbar to switch to the Edit mode.

To change the state of a pulse:

1. In the Waveform Editor, select a pulse by clicking anywhere within the pulse.
2. In the Pulse Properties view, select the desired state in the State box. If you are working with a bus pulse, you can select **Set Value** in the State box, and then enter the value in the Value box below it. The value entered in the Value box must be in hexadecimal format.

   Changing a pulse from Low to High or High to Low effectively removes it from the waveform, since it now has the same value as the pulses on either side.

To change the duration of a pulse:

1. In the Waveform Editor, select a pulse by clicking anywhere within the pulse.

2. In the Pulse Properties view, click inside the Duration edit box and type a time value.

   You don't have to enter the time unit. The Waveform Editor automatically appends the time unit specified in the Time Settings dialog box to the time value. To apply a different time unit, see "Setting Simulation Time Unit" on page 342.

Repeating Pulses

You can use the Pulse Properties view to repeat successive pulses in a waveform. The pulses at the end of a waveform can be repeated forever in the Waveform Editor.

Note

- A single pulse cannot be repeated. To repeat pulses, you must select more than two successive pulses.
- Pattern pulses cannot be repeated forever.

To repeat pulses:

1. Be sure you are in the Edit operation mode. If not, choose **Edit > Edit Mode** or click the icon on the toolbar to switch to the Edit mode.
2. In the Waveform Editing area, select the pulses that you want to repeat by dragging the cursor across the waveform to cover several successive pulses. The selected pulses are marked with thick black lines above and below them.
3. In the Pulse Properties view, do either of the following:
   - In the Repeat edit box, type any integer larger than 1, and then press **Enter**.

     The selected pulses are repeated by the specified multiple.
   - If the selected pulses are located at the end of a waveform, you can select **Forever** to repeat the selected pulses forever.

   Note

   The simulation time for the Repeat Forever waveform can be controlled by the run time or conditional settings in a simulator.

Scaling Patterns and Waveforms

You can use the Pulse Properties view to make any pattern, pulse(s), or an entire waveform longer by scaling it.

To scale patterns and waveforms:

1. Be sure you are in the Edit operation mode. If not, choose **Edit > Edit Mode** or click the icon on the toolbar to switch to the Edit mode.

2. In the Waveform Editing area, select the pulse(s) that you want to scale by clicking anywhere within the pulse or dragging the cursor across the waveform to cover several successive pulses. The selected pulses are marked with thick black lines above and below them.

   If you want to scale an entire pattern or waveform, select all its pulses.

3. In the Pulse Properties view, click inside the Scale edit box.

4. Type any integer larger than 1, and press **Enter**.

   The length of the selection is multiplied by this value. (A pulse is multiplied only if it is not part of a Repeat Forever waveform.)

   If a complete waveform contains a pattern, that pattern is scaled along with the rest of the waveform. (The original definition of the pattern is unaffected by the scaling.)

See Also ▶"Selecting Waveforms or Pulses" on page 350

▶ "Graphically Changing the Length of a Pulse" on page 350

Setting Waveform Polarity and Offset

The Waveform Properties view allows you to change waveform polarity and add an offset at the beginning of a waveform.

You cannot change waveform polarity or offset for patterns.

To display the Waveform Properties view, click the **Waveform** tab at the bottom of the right-hand pane. If the Waveform tab is not displayed, choose **View > Waveform Properties View**.

Note

You cannot change waveform properties for individual waveforms included in a bus. However, you can set properties for the bus waveform.

To set waveform properties:

1. In the Waveform Editing area, select the waveform name for which you want to change properties.

   The Waveform Properties view displays the waveform details in the Wave Data box, including the state and duration for each pulse in the selected waveform.

2. In the Waveform Properties view, do the following:

   - To change polarity for the selected waveform, select a different polarity in the Polarity box.

     The letter behind the waveform name changes to indicate the updated polarity.

   - To add an offset at the beginning of the waveform, enter the offset value in the Offset box. The offset value must be a number greater than 0.

The waveform shifts to the right if the offset value was increased or to the left if the offset value was decreased.

Deleting, Cutting, Copying, Pasting, and Duplicating Pulses or Waveforms

Pulses or waveforms that can be selected (a single pulse, multiple successive pulses, or the entire waveform) can be deleted, cut, or copied to the clipboard for pasting elsewhere.

To delete, cut, or copy pulses or waveforms:

1. Select the waveform or pulse(s) that you want to edit.
2. Do any of the following:
   - Choose **Edit > Clear**.

     The selected waveform or pulse(s) is removed, but is not copied to the clipboard.
   - Choose **Edit > Remove Wave**.

     The Remove Waves dialog box opens. In the Wave List, select one or more waveforms to be deleted and click **Remove**. You can use the Shift or Ctrl key to select multiple waveforms or select and remove them one at a time.
   - To cut a waveform or pulse(s), choose **Edit > Cut**.

     The selected waveform or pulse(s) is removed, and is copied to the clipboard.
   - To copy a waveform or pulse(s), choose **Edit > Copy**.

     The selected waveform or pulse(s) is copied to the clipboard without being removed.

To paste the copied waveform or pulse(s) into the current waveform:

1. Select a single pulse, multiple successive pulses, or an entire waveform.
2. Choose **Edit > Paste**.

   The data in the Clipboard will replace the selected waveform or pulse(s).

To duplicate a waveform:

1. Select the waveform name that you want to duplicate.
2. Choose **Edit > Duplicate Wave**.

   The software adds an exact copy of the selected waveform to the display, and adds “_0” to the waveform name.

See Also ▶“Selecting Waveforms or Pulses” on page 350

Saving Waveforms

The first time you save a waveform file, the Waveform Editor creates a *project*.wdl file. The .wdl file contains the waveforms in WDL (waveform description language) format. You can use the Save As command to save the file under a different base name to create multiple stimulus files for a single project.

Caution

You can perform an unlimited number of Undos and Redos until you save the file. At that point, the files are updated and all Undo/Redo information is lost. Do not save the file if there are still changes you want to Undo or Redo.

The Waveform Editor can automatically save the file you are working on. You can use the Options dialog box to select an auto-save option.

To set auto-save options:

1. In the Diamond window, choose **Tools > Options**.

   The Options dialog box opens.

2. In the left-hand pane, select **Waveform Editor**.
3. In the right-hand pane, under Auto Save, select **1 Minute**, **5 Minutes**, or **10 Minutes** to let the system automatically save your file every 1, 5, or 10 minutes. Or select **Never Save** to cancel the auto-save function.
4. Click **OK**.

Showing Waveforms

Sometimes waveforms have been hidden from view. You can restore the view of hidden waveforms using the Show command.

To show a hidden waveform in the Waveform Editor:

1. Choose **Edit > Show Wave**.

   The Show Waves dialog box opens. The Wave List box displays all the available signals. You can use the Filter options on the left to narrow the signal list.

2. In the Wave List, select the name of the signal you want to show. You can use the Shift or Ctrl key to select multiple signal names.
3. Click **Show**.

   The software restores the view of the selected waveform.

Hiding Waveforms

If you decide not to display a particular waveform, you can temporarily hide it without deleting it from the Waveform Editor database.

To hide a waveform:

1. In the waveform Editing area, select the name of the waveform you want to hide.

   The Waveform Editor highlights the waveform name.

2. Choose **Edit > Hide Wave**.

   The Hide Wave command hides the highlighted waveform name and its waveform.

Viewing Waveform and Pulse Properties

When you are in the Edit mode, you can select a waveform or a pulse in the Waveform Editing area and view its properties.

Pulse properties like pulse state, duration, and scale are displayed in the Pulse Properties view. Waveform data, polarity, and offset are displayed in the Waveform Properties view.

To view pulse information in the Pulse Properties view:

1. Make sure you are in the Edit mode. If not, choose **Edit > Edit Mode** or click the icon on the toolbar to switch to the Edit mode.
2. At the bottom of the right-hand pane, click the **Pulse** tab to display the Pulse Properties view.

   Note

   If the Pulse tab is not available, choose **View > Pulse Properties View** to display it.

3. In the Waveform Editing area, select a pulse.

   The selected pulse is marked with thick black lines above and below it. The pulse information will appear in the Pulse Properties view.

To view waveform data in the Waveform Properties view:

1. At the bottom of the right-hand pane, click the **Waveform** tab to display the Waveform Properties view.

   Note

   If the Waveform tab is not available, choose **View > Waveform Properties View** to display it.

2. In the Waveform Editing area, select any waveform name on the left.

The selected waveform name is highlighted. The waveform data will appear in the Waveform Properties view.

Using a Marker

A marker is a red vertical line on the waveform time line. It is useful as a reference point to query the simulation data.

To place a marker:

1. In the Waveform Editing area, click the place on the time line where you want to insert a marker. You can use the Jump to commands on the View menu to locate the exact place.

   A query bar (a black vertical line) appears.

2. Choose **View > Place Marker**.

   A marker is inserted at the current location of the query bar. You may not see the red marker because it is covered by the black query bar. Click anywhere else to move the query bar, and then you will see the red marker.

To move the query bar to the marker:

1. In the Waveform Editing area, click anywhere on the time line to position a query bar.
2. Choose **View > Jump to > Marker**.

   The query bar is re-positioned at the marker.

   This is useful when you are displaying the time line at high magnification and you would have to press the Left or Right keys many times to return to the marker.

To hide a marker:

- In the Waveform Editor, choose **View > Hide Marker**.

  The marker is removed from the waveform display.

Positioning the Query Bar with the Jump to Commands

A query bar is a black vertical line on the waveform time line used to query the simulation data. You can use the Jump to commands on the View menu to position the query bar to a specific time.

To position a query bar:

1. In the Waveform Editing area, click anywhere on the time line to display a query bar.

2. Choose **View > Jump to**, and then select the required command to position the query bar to the specific time.

   The below table describes each Jump to command in detail.

Table 18:

Command Name	Description	Keyboard Shortcut
Next Change	Moves the query bar to the next transition in the currently selected signal. If the currently selected signal is a bus, the query bar moves to the next transition of any signal in the bus. This command is typically used to traverse the simulation data. Starting at a primary input (which is the most common activity initiator), you can jump from one initiating event to another. You can find later events on other signals that result from the changes on the primary input.	+
Marker	Positions the query bar at the marker. The principal use of this command is simply to return the query bar to the marker. This is especially useful when you are displaying the time line at high magnification and you would have to press Shift+Left or Shift+Right many times to return to the marker. For information on how to place a marker, see Figure on page 357.	-
Time...	Opens a dialog box that allows you to move the query bar to a specified time.	
Time=0	Moves the query bar to time=0. This is the left-most point on the time axis of the waveform.	Home
Time=End	Moves the query bar to the last time point of the simulation. This is the right-most point on the time axis of the waveform.	End
Tick Left	Moves the query bar one tick mark to the left (“backwards” in time). The interval between tick marks changes as you change the time scale with the Zoom commands.	Left
Tick Right	Moves the query bar one tick mark to the right (“forward” in time). The interval between tick marks changes as you change the time scale with the Zoom commands.	Right
10 Left	Moves the query bar one major tick mark to the left (“backwards” in time). The interval between tick marks changes as you change the time scale with the Zoom commands.	Shift+Left
10 Right	Moves the query bar one major tick mark to the right (“forward” in time). The interval between tick marks changes as you change the time scale with the Zoom commands.	Shift+Right

Changing the Waveform Display Options

You can use the Options dialog box to control the following waveform display options.

- **Bus Radix** – Controls how bus values are displayed in the Waveform Editor.

- **Show Grid Lines** – Shows or hides grid lines in the Waveform Editor.
- **Color** – Specifies the Waveform Editor color scheme.

To change the waveform display options:

1. In the Diamond window, choose **Tools > Options**.

   The Options dialog box opens.
2. In the left-hand pane, under Waveform Editor, select **General**.
3. In the right-hand pane, specify the Bus Radix and Show Grid Lines options as you want.
4. In the left-hand pane, under Waveform Editor, select **Color**.

   The current color settings are shown in the right-hand pane.
5. Click on the current color to open the Select Color dialog box, select the desired color, and click **OK**.
6. Click **OK** to apply your settings and close the Options dialog box.

Exporting the Stimulus File

The Waveform Editor saves the waveforms you create in it as a .wdl file. You can export (create) a Verilog (.tf) or VHDL (.vht or .tb) stimulus file from a .wdl file at any time. The Waveform Editor translates the waveform database in the .wdl file into the stimulus format in the appropriate output files. You can then use the exported stimulus file to simulate your design in a simulator.

To export the stimulus file:

1. In the Waveform Editor, choose **File > Export**.
2. In the Export Waveform File dialog box, do the following:
   - Click **Browse** to specify the existing .wdl file which includes the waveform information you want to export.
   - Click **Browse** to specify the name and location of the output file.
   - In the Export Type box, select an export format. This appends the appropriate file extension to the output file.
3. Click **OK**.

   The Waveform Editor exports the stimulus file to the specified location.

Chapter 6how the

Applying Design Constraints

Constraints are instructions applied to design elements to guide the design toward desired results and performance goals. They are critical to achieving timing closure or managing reusable intellectual property (IP). The most common constraints are those for timing and pin assignment, but constraints are also available for placement, routing, and many other functions.

Constraints can include timing constraints defined in synthesis constraint files (SDC) or HDL attributes. SDC is used for the synthesis tool. Post-synthesis constraints known as preferences can also be specified. The flow combines these together. All three sources of constraints specify design goals. Synthesis, map, and place-and-route work to meet these goals. Timing analysis reports whether or not the goals were met.

See Also

- "Constraints Reference Guide" on page 1194
- "Multiple Entry Constraint Flow" on page 360
- "Using Preferences" on page 375
- "Setting Preferences" on page 459
- "Migrating Pin Assignments" on page 484
- "Integrated Synthesis" on page 583
- "Achieving Timing Closure" on page 571
- "Lattice Synthesis Engine (LSE) Constraints" on page 1339

Multiple Entry Constraint Flow

Constraints from mutiple sources can be used and modified in Lattice Diamond in multiple ways, such as: timing constraints from SDC or LDC synthesis files; constraints defined in HDL source files; or with post-synthesis constraints known as logical preferences that are defined in the logical preference file (.lpf).

You can set constraints using one or more of the following methods:

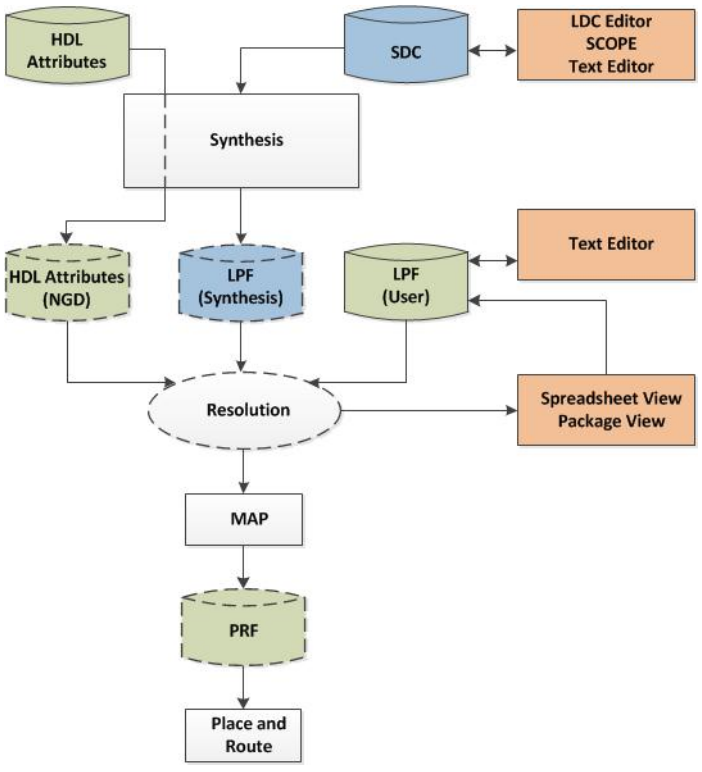
- You can assign HDL attributes in the HDL or schematic source files. After synthesis and translation, these defined attributes can be viewed and modified in Diamond's preference views, and the modifications can be saved to the logical preference file.

- If you are using the Lattice Synthesis Engine, you can assign SDC constraints using the Lattice Design Constraints (LDC) Editor and save them to an .ldc constraints file. If you are using Synplify Pro, you can assign SDC constraints using the SCOPE editor tool and save them to an .sdc file.
- You can apply SDC constraints added in Diamond by enabling the new "Use LPF Created from SDC in Project" strategy option. Refer to "Using SDC Constraints in Place and Route" on page 365.
- You can assign logical preferences in the logical preference file (.lpf) using a text editor or Diamond's preference-editing views: Device View, Spreadsheet View, Package View, Netlist View, NCD View, and Floorplan View.
- You can use Spreadsheet View to modify timing constraints that have been defined in Synplify Pro or Lattice Synthesis Engine (LSE) and save the modified constraints as preferences to the active .lpf file.

The constraints entered from multiple entry points are resolved in the Diamond software. Refer to "Resolving SDC Constraints with Other Constraints from Multiple Flow" on page 365.

The following figure shows a high-level flow of how constraints from mutiple sources can be used and modified in Lattice Diamond.

Figure 49:



See Also "Constraints Reference Guide" on page 1194

- "Using SDC Constraints" on page 364
- "Using Preferences" on page 375
- "Adding FPGA Attributes to HDL" on page 569

Constraint Implementation

The following steps illustrate how you might assign constraints and implement them at each stage of the design flow, using synthesis constraints, logical preferences and timing analysis.

1. Define the constraints in the pre-synthesis stage using one or both of the following methods:
   - Define constraints as attributes within the HDL or schematic model.
   - Define timing constraints using Synplify Pro or Lattice Synthesis Engine (LSE).
2. Synthesize and translate the design.

   When Synplify Pro is used, the synthesis process translates the design into an EDIF netlist. Run the **Translate Design** process to translate the EDIF netlist into a binary logical native generic database (.ngd) file, which will contain all HDL attributes.

   When LSE is used, the synthesis process both synthesizes the design and translates it into an .ngd file.

   The synthesis process writes timing constraints that were defined in Synplify Pro or LSE to a synthesis .lpf file in the project directory. The synthesis .lpf will not appear in Diamond, but the timing constraints will be displayed in Diamond's Spreadsheet View and color-coded as synthesis values.
3. Open one or more of the following views to define new constraints, called preferences, or to modify existing constraints from the source files and save them as preferences:

   Spreadsheet View – Modify timing constraints that were defined in the synthesis tool and save them as logical preferences to the active.lpf file. Set timing objectives such as fMAX and I/O timing. Define signaling standards and make pin assignments. Assign clocks to primary or secondary routing resources. Set parameters for simultaneous switching outputs and perform SSO analysis. Define groups of ports, cells, or ASIC blocks. Create UGROUPs from selected instances to guide placement and routing. Establish REGIONs for UGROUPs or to reserve areas of the floorplan. Run PIO design rule checking.

   Device View – Examine FPGA device resources. Reserve sites that should be excluded from placement and routing.

   Netlist View – View the design tree by ports, instances, and nets. Assign pins for selected signals. Set timing constraints. Define groups from selected ports or registers. Create UGROUPs from selected instances to guide placement and routing.

Package View – View the pin layout of the design. Modify signal assignments and reserve pin sites that should be excluded from placement and routing. Examine the status of SSO pins. Run PIO design rule checking.

Floorplan View – View the device layout. Draw bounding boxes for UGROUPs. Draw REGIONs for the assignment of groups or to reserve areas. Reserve sites and REGIONs that should be excluded from placement and routing. Run PIO design rule checking.

4. Save the preferences to the logical preference file (.lpf).
5. Run the **Map Design** process (map).

   This process reads the .ngd and .lpf files and produces a native circuit description (.ncd) file and a physical preference (.prf) file. The .prf file is derived from preference-related .ngd attributes and the .lpf file. Non-preference-related .ngd attributes are written directly into the .ncd file.

   The .prf file is an internal file generated by the Map engine that contains preferences used by the PAR engine. It should not be edited, because the changes will be lost when it is regenerated.
6. Run the **Map TRACE** process and examine the timing analysis report. Modify preferences as needed and save them.
7. Run the **Place & Route Design** process (par).

   This process reads the post-MAP .ncd file and the .prf file, and appends placement and routing to a post-PAR .ncd file.
8. Open one of the following views, as desired, to examine timing and placement and create new UGROUPs.

   Timing Analysis View – Examine details of timing paths. Cross-probe selected paths to Floorplan and Physical Views. Create one or more timing preference files (.tpf) and experiment with sets of modified preferences for the purpose of timing analysis, using the TPF Spreadsheet View. Select and export the desired timing preferences from TPF Spreadsheet View to the regular Spreadsheet View. From Spreadsheet View, save the timing preferences to the .lpf file.

   NCD View – Examine placement assignments. Create new UGROUPs, as needed, from selected instances.
9. Cross-probe to **Logic Block View** from NCD View, Floorplan View, or Physical View, to view logic details of a selected component.
10. Modify preferences or create new ones using any of the preference-editing views. Save the preference changes and rerun the **Place & Route Design** process.
11. Repeat Steps 3 through 10 until timing objectives are met.

You can repeat the flow using a new implementation of your design with a different set of preferences. See "Working with Run Manager" on page 94 and "Managing Project Sources" on page 34.

Note

Refer to Importing ispLEVER Projects to ensure the compatibility of designs that use the ispLEVER constraint data flow.

See Also
- "Preference Flow" on page 376
- "Preference-Editing Views and Memory" on page 407
- "Using Diamond's Preference Views" on page 406
- "Analyzing Static Timing" on page 694
- "Achieving Timing Closure" on page 571

Using SDC Constraints

You can enter constraints in the synthesis design constraints (SDC) file using one of the following two synthesis tools:

- Lattice Synthesis Engine (LSE) LDC Editor

  For more information about entering constraints with LDC Editor, refer to "Applying Lattice Synthesis Engine Constraints" on page 366.

- Synplify Pro SCOPE Constraints Editor

  For more information about entering constraints with Synplify Pro SCOPE Constraints Editor, in the Synplify Pro online help, refer to "SCOPE Constraints Editor" help.

During synthesis, a read-only logical preference file (.lpf) is generated, which can be included and specified as the active preference file in your Diamond project.

The following Synopsis Design Constraints can be set in the SDC:

- "create_clock" on page 1341
- "create_generated_clock" on page 1343
- "set_clock_groups" on page 1344
- "set_false_path" on page 1345
- "set_input_delay" on page 1346
- "set_max_delay" on page 1347
- "set_min_delay" on page 1347
- "set_multicycle_path" on page 1348
- "set_output_delay" on page 1348

For more information about these constraints, refer to "Synopsys Design Constraints (SDC)" on page 1339.

Constraints entered from multiple entry points are resolved in the Diamond software. Refer to "Resolving SDC Constraints with Other Constraints from Multiple Flow" on page 365.

See Also "Using SDC Constraints in Place and Route" on page 365

- "Conflicting HDL Attributes, Constraints, and Preferences" on page 401
- "Using Diamond's Preference Views" on page 406

Using SDC Constraints in Place and Route

When you chose the "Use LPF Created from SDC in Project" strategy setting, constraints set in Lattice Synthesis Engine (LSE) LDC Editor or Synplify Pro SCOPE Constraints Editor will be applied during the Diamond software Place & Route process.

To use SDC constraints in Place & Route

1. Specify "Use LPF Created from SDC in Project" in your project's strategy. Refer to "Specifying Strategy Options" on page 52.

   Note

   Refer to the "Strategy Reference Guide" on page 1112 for more information about LSE and Synplify Pro versions of the "Use LPF Created from SDC in Project" strategy.

2. Run the Synthesize the Design process. Refer to "Running Processes" on page 70.
3. Set the new LPF file as the active file. Refer to "Managing Constraint Files" on page 41.
4. Run the Place & Route Design process. Refer to "Running Processes" on page 70.

See Also "Using SDC Constraints" on page 364

Resolving SDC Constraints with Other Constraints from Multiple Flow

When you have multiple sources of constraints, the Diamond software follows specific rules to resolve conflicting constraints. Refer to "Preference Conflict Resolution" on page 395.

See Also "Multiple Entry Constraint Flow" on page 360

Applying Lattice Synthesis Engine Constraints

Diamond's Lattice Synthesis Engine (LSE) enables you to set Synopsys® Design Constraints (SDC), which are directly interpreted by the synthesis engine. When you use LSE, these SDC constraints are saved to a Lattice Design Constraints file (.ldc). You can create several .ldc files and select one of them to serve as the active synthesis constraint file for an implementation. You can also cause a synthesis preference file to be generated when the design is synthesized. The synthesis preferences can then be merged with the logical preference file (.lpf).

Lattice Design Constraints (LDC) Editor, as well as Source Editor, are available for creating and editing .ldc files. LDC Editor provides a spreadsheet style user interface that enables you to quickly create and edit Synopsys Design Constraints.

See "Synopsys Design Constraints (SDC)" on page 1339 for descriptions of the SDC constraints that are supported by the Lattice Synthesis Engine and LDC Editor.

See Also

- "Defining Synthesis Constraints Using LDC Editor" on page 366
- "Creating a New LDC Synthesis Constraint File" on page 367
- "Integrated Synthesis" on page 583
- "Optimizing LSE for Area and Speed" on page 581

Defining Synthesis Constraints Using LDC Editor

LDC Editor presents the contents of a Lattice Design Constraints File (.ldc) in a spreadsheet style format. You can use LDC Editor to open and edit a current .ldc file or create a new one. Individual sheets are provided for defining clocks, setting input and output delays, and defining delay paths for multicycle, minimum and maximum delays, and false paths.

By default, LDC Editor is launched when you create a new .ldc file. Each sheet of LDC Editor allows you to define synthesis constraints by double-clicking a cell and selecting or typing a value.

Each row includes an Enabled check box, which is selected by default. If you clear this check box, the constraint is ignored by LSE. This can be an easy way to test different settings.

See Also

- "Creating a New LDC Synthesis Constraint File" on page 367
- "Defining Clocks Using LDC Editor" on page 369
- "Setting Input and Output Delays Using LDC Editor" on page 369
- "Defining Delay Paths Using LDC Editor" on page 370
- "Defining Clock Groups in LDC Editor" on page 371
- "Defining Generated Clocks in LDC Editor" on page 371

- "Setting Attributes in LDC Editor" on page 372
- "Generating a Synthesis Preference File" on page 374
- "Synopsys Design Constraints (SDC)" on page 1339

Running LDC Editor

Run LDC Editor by opening or creating an .ldc file for an implementation.

Note

In order to open or create an .ldc file, LSE must be the selected synthesis tool for the project. Choose **Project > Synthesis Tools** and select **Lattice LSE**.

You can have multiple .ldc files for an implementation, but only one can be opened in LDC Editor at a time.

To run LDC Editor, do one of the following:

- Double-click an .ldc file name in the Synthesis Constraints Files folder in the File List view.
- If you have set Source Editor to be the default program for .ldc files, right-click the .ldc file name in the File List view and choose **Open With**. In the Open With dialog box, choose **LDC Editor** and click **OK**.
- If no .ldc file exists yet for the active implementation, create an .ldc file as described in Creating a New LDC Synthesis Constraint File.

See Also
- "Managing Constraint Files" on page 41
- "Using Source Editor for LDC Files" on page 373

Creating a New LDC Synthesis Constraint File

Before you create a new .ldc file, verify that Lattice Synthesis Engine has been selected as the synthesis tool. In the Processes tab, "Lattice Synthesis Engine" should be listed under "Synthesize Design." If it is not listed, choose **Project > Synthesis Tool**, and select **Lattice LSE**.

To make sure that the LDC file opens in LDC Editor instead of Source Editor, verify that LDC Editor has been selected as the default program in Tool > Options > Environment > File Associations.

To create a new .ldc synthesis constraint file:

1. Choose **File > New > File**.

   Alternatively, right-click the **Synthesis Constraint Files** folder and choose **Add > New File**.

2. In the New File dialog box, select **Source Files** from the Categories list. Select  **LDC Files** from the Source Files list.

3. Type a name for the new .ldc file and specify the directory location.
4. Select the **Add to Implementation** option if you want to use the file with the currently active implementation.
5. Click **New**.

   LDC Editor opens and displays the spreadsheet and three tabs for creating and editing synthesis constraints. If you selected the Add to Implementation option, the new .ldc file will appear in the Synthesis Constraint Files folder.

   Drop-down menus of source elements are provided when you double-click the Clock Source cells or the Inputs/Outputs Ports cells.

   Note

   If manual compilation has been selected as the default, the lists of source elements might not be displayed. Choose **Design > Compile** to display them.

6. Enter or edit values for clocks, inputs and outputs, and delay paths:
   a. Double-click a cell and type a new value, or select a value from the drop-down list.
   b. Use the arrow keys or the Tab key to move to another cell and select it for editing.
   c. To add a new row, right-click inside a row and choose **Insert Row** from the pop-up menu.
7. Choose **File > Save**.

See Also

- "Defining Clocks Using LDC Editor" on page 369
- "Setting Input and Output Delays Using LDC Editor" on page 369
- "Defining Delay Paths Using LDC Editor" on page 370
- "Synopsys Design Constraints (SDC)" on page 1339
- "Managing Constraint Files" on page 41
- "Integrated Synthesis" on page 583
- "Optimizing LSE for Area and Speed" on page 581

Setting Design Rule Checks for LDC Editor

LDC Editor runs a design rule check (DRC) to verify that the constraints you enter are legitimate. You can control when these checks happen. By default, the DRC happens before saving the .ldc file. You can also set it to happen while you are creating the constraints for immediate feedback. The DRC marks any problems and gives you a chance to correct them.

To set when LDC Editor runs DRC:

1. Choose **Tools > Options**.

   The Options dialog box opens.

2. In the left frame, choose **LDC Editor**.
3. If you want to run DRC:
   - before saving the .ldc file, select **Run DRC check before saving**.
   - while creating constrains, select **Enable realtime DRC check**.

   If you clear both options, the DRC never runs. In that case, you may be saving incomplete or inaccurate constraints.
4. Click **OK**.

Defining Clocks Using LDC Editor

The Clocks tab of LDC Editor enables you to define an alias to be associated with an existing clock port or net from the source file. You can also define clocks by dragging-and-dropping from Netlist Analyzer into LDC Editor.

To define a clock using LDC Editor:

1. Double-click the **Source** cell to select an existing clock pin, port, or net.
2. If desired, enter an alias for the clock in the Clock Name cell.
3. Enter a clock period in nanoseconds in the Period(ns) cell.
4. If desired, you can specify the duty cycle. Double-click the Waveform Low Edge cell and enter the length of the high portion of the cycle in nanosecond. The default duty cycle is 50%. The waveform starts with the rising edge.

To define a clock in LDC Editor using Netlist Analyzer:

1. Start Netlist Analyzer. Refer to "About Netlist Analyzer" on page 58.
2. Select desired clock object (port or net) from Netlist Analyzer and drag-and-drop into Clock Name cell of the LDC Editor Clocks tab.
3. If desired, enter an alias for the clock in the Clock Name cell.
4. Enter a clock period in nanoseconds in the Period(ns) cell.
5. If desired, you can specify the duty cycle. Double-click the Waveform Low Edge cell and enter the length of the high portion of the cycle in nanosecond. The default duty cycle is 50%. The waveform starts with the rising edge.

See Also
- "create_clock" on page 1341
- "Creating a New LDC Synthesis Constraint File" on page 367
- "About Netlist Analyzer" on page 58

Setting Input and Output Delays Using LDC Editor

The Inputs/Outputs tab of LDC Editor enables you to specify input and output delays relative to a clock. You can also define input and output delays by dragging-and-dropping from Netlist Analyzer into LDC Editor.

To set an input or output delay using LDC Editor:

1. Double-click the **Type** cell to select the type of delay (input or output).
2. Select from the input or output ports in the Port cell.
3. In the Clock cell, select an existing clock or an alias name that has been defined for a clock.
4. Select either Max or Min or neither. Do not select both.
   - Max means that the delay value refers to the longest path.
   - Min means that the delay value refers to the shortest path.
   - Neither means that the maximum and minimum delays are assumed to be equal.
5. Enter a delay value in nanoseconds in the Value(ns) cell.

To set an input or output delay in LDC Editor using Netlist Analyzer:

1. Start Netlist Analyzer. Refer to "About Netlist Analyzer" on page 58.
2. Select from the input or output ports from Netlist Analyzer and drag-and-drop into the Port cell of the LDC Editor Input/Outputs tab.
3. In the Clock cell, select an existing clock or an alias name that has been defined for a clock.
4. Select either Max or Min or neither. Do not select both.
5. Enter a delay value in nanoseconds in the Value(ns) cell.

See Also

- "Defining Clocks Using LDC Editor" on page 369
- "set_input_delay" on page 1346
- "set_output_delay" on page 1348
- "Creating a New LDC Synthesis Constraint File" on page 367
- "About Netlist Analyzer" on page 58

Defining Delay Paths Using LDC Editor

The Delay Paths tab of LDC Editor enables you to define a Multicycle path, specify a Max_Delay or a Min_Delay for a timing path, and identify a False path that is to be excluded from timing analysis. You can also perform the same functions by dragging-and-dropping from Netlist Analyzer into LDC Editor.

To define a delay path using LDC Editor:

1. Double-click the **Delay Type** cell and select the type of delay.
2. Specify the path information, delay, and cycles as appropriate for the selected delay type.

To define a delay path in LDC Editor using Netlist Analyzer:

1. Start Netlist Analyzer. Refer to "About Netlist Analyzer" on page 58.
2. Select object (port, net, or register) from Netlist Analyzer and drag-and-drop into the From or To cell of the LDC Editor Delay Paths tab.
3. Specify the path information, delay, and cycles as appropriate for the selected delay type.

See Also
- "set_multicycle_path" on page 1348
- "set_max_delay" on page 1347
- "set_min_delay" on page 1347
- "set_false_path" on page 1345
- "Creating a New LDC Synthesis Constraint File" on page 367
- "About Netlist Analyzer" on page 58

Defining Clock Groups in LDC Editor

The Clock Groups tab of LDC Editor enables you to define clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during timing analysis.

To define clock groups using LDC Editor:

1. Double-click the **Type** cell and select the type of clock group.
2. Specify the group information as appropriate for the selected clock group.

See Also
- "set_clock_groups" on page 1344
- "Creating a New LDC Synthesis Constraint File" on page 367

Defining Generated Clocks in LDC Editor

The Generated Clocks tab of LDC Editor enables you to define internally generated clocks. You can also perform the same functions by dragging-and-dropping from Netlist Analyzer into LDC Editor.

To define generated clocks using LDC Editor:

1. In the **Source** cell, select the source clock for the generated clock.
2. Specify the other values as appropriate for the generated clock.

To define generated clocks in LDC Editor using Netlist Analyzer:

1. Start Netlist Analyzer. Refer to "About Netlist Analyzer" on page 58.
2. Select the source clock from Netlist Analyzer and drag-and-drop into the Source cell of the LDC Editor Generated Clocks tab.

3. Specify the other values as appropriate for the generated clock. You can also specify the Master Clock and Object values by dragging from Netlist Analyzer.

See Also ▶"create_generated_clock" on page 1343

▶ "Creating a New LDC Synthesis Constraint File" on page 367

Setting Attributes in LDC Editor

The Attributes tab of LDC Editor enables you to specify Synplify Lattice Attributes that are supported by the Lattice Synthesis Engine (LSE). You can also define objects by dragging-and-dropping from Netlist Analyzer into LDC Editor.

To set attributes using LDC Editor:

1. Double-click the **Attribute** cell and select the desired attribute.
2. Specify object type, object, value type, and value, as appropriate for the attribute.

To set attributes in LDC Editor using Netlist Analyzer:

1. Start Netlist Analyzer. Refer to "About Netlist Analyzer" on page 58.
2. In LDC Editor Attribute tab, double-click the **Attribute** cell and select the desired attribute.
3. Specify object type and value, as appropriate for the attribute.
4. Select object (port, net, or register) from Netlist Analyzer and drag-and-drop into the Object cell of the LDC Editor Attribute tab.

Note

You can also drag-and-drop an object from Netlist Analyzer to the Object cell of LDC Editor Attribute tab before you specify the attribute. Object type will then be selected automatically. You can then specify attribute and value.

See Also ▶"Lattice Synthesis Engine-Supported HDL Attributes" on page 1349

▶ "Creating a New LDC Synthesis Constraint File" on page 367

▶ "About Netlist Analyzer" on page 58

Compiling the Design Using LDC Editor

By default, LDC Editor is set to compile the design automatically. With automatic compilation, LDC Editor automatically performs checks as to whether the design data is outdated. This happens whenever LDC Editor is launched or activated. If the design data is found to be outdated, the compilation will begin at once.

You can change this default setting to manual compilation, which will cause LDC Editor to check the data only when the Compile command is used.

To change to manual compilation:

1. Choose **Tools > Options**.
2. Select **LDC Editor** from the list of tools.
3. In the “Run LSE synthesis compile step” options, select **Manually**.
4. If you want the editor to run a check for any rule violations each time you save the .ldc file, select **Run DRC check before saving**.
5. Click **OK**.

To compile the design manually:

- Choose **Design > Compile** or click the Compile button on the toolbar.

  LDC Editor checks the design data. If the data is found to be outdated, it begins the compilation.

See Also ▶“Integrated Synthesis” on page 583

Using Source Editor for LDC Files

Any .ldc file that you have created using LDC Editor can be viewed and edited in Source Editor. Likewise, an .ldc file that you have created in Source Editor can be viewed and edited in LDC Editor.

One advantage of using Source Editor is that it allows you to open multiple .ldc files at once, whereas LDC Editor will load only one .ldc file at a time.

To open an .ldc file in Source Editor:

1. In the File List view, expand the Synthesis Constraints folder.
2. Right-click the name of the .ldc file you want to open and choose **Open With**.
3. In the dialog box, select **Source Editor** and click **OK**.

   The .ldc file opens in Source Editor window as an ASCII file.

See Also ▶“Changing the Default Editor for LDC Files” on page 373

Changing the Default Editor for LDC Files

Diamond gives you the option of setting Source Editor or LDC Editor as the default program for .ldc files.

To change the default editor for .ldc files:

1. Choose **Tools > Options**.

2. In the Environment section of the Options dialog box, select **File Associations**.
3. In the Extensions column of File Associations, scroll down to **ldc**.
4. In the Default Programs column, choose **Source Editor** or **LDC Editor** from the drop-down menu.

   The next time you create or select an .ldc file, it will open in the editor you selected as the default.

See Also ▶"Managing Constraint Files" on page 41

Optimizing for Timing

The SDC constraints that you set in the .ldc file will drive optimization of the design if you set the optimization goal for timing in the active strategy file.

To set the optimization goal for timing:

1. From the File List view, expand the Strategies folder and double-click the name of the active strategy.
2. In the Strategies dialog box, select **LSE** from the Synthesis Design folder in the Process pane.
3. In the LSE pane on the right, scroll down to Optimization Goal and select **Timing** in the Value column.
4. Click **OK**.

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

▶ "Optimizing LSE for Area and Speed" on page 581

Generating a Synthesis Preference File

You have the option of automatically generating a synthesis preference file for an implementation when you synthesize the design. The generated synthesis file includes preferences that are equivalent to those in the .ldc file. These synthesis preferences can later be merged with those of the active .lpf file.

To automatically generate a synthesis preference file:

1. From the File List view, expand the Strategies folder and double-click the name of the active strategy.
2. In the Strategies dialog box, select **LSE** from the Synthesis Design folder in the Process pane.
3. In the LSE pane on the right, scroll down to Output Preference File and select **True** in the Value column.

See Also ▶"Integrated Synthesis" on page 583

Adding an LDC File to an Implementation

You can add multiple .ldc files to an implementation and view each of them in LDC Editor.

To add an existing .ldc file to an implementation:

1. In the File List view, right-click the Synthesis Constraints Folder and choose **Add > Existing File**.
2. Browse to the directory that contains the desired .ldc file, select the file, and click **Add**.

   The file is added to the implementation and is displayed in the Synthesis Constraint Files list.

See Also ▶"Managing Constraint Files" on page 41

Activating an LDC File

To apply the synthesis constraints, you need to identify an .ldc file as the active one for the implementation.

To activate an .ldc file for an implementation:

- Right-click the file name and choose **Set as Active LDC**.

  By default, the active .ldc file is compiled when you synthesize the design.

See Also ▶"Managing Constraint Files" on page 41

Using Preferences

Preferences are constraints that guide the design toward performance goals. They can be created and edited using Diamond's six preference-editing views, or they can be created and edited manually in the logical preference file (.lpf) using a text editor. Preferences, unlike HDL attributes, are written to the .lpf file.

This section describes the major types of preferences and the role they play in the design flow. For complete descriptions of all preferences, see the "Constraints Reference Guide" on page 1194.

The Logical Preference File

The logical preference file (.lpf) is your source file for storing constraints. It is the constraint file used as input for developing and implementing the design. A new .lpf file is automatically generated when you create a new project. It is assigned the project file name by default and set as the active constraint file for the project's initial implementation.

The Map Design process depends on the .lpf file. If you map the design and then modify the active .lpf file, the map process and any downstream processes must be rerun.

You can create and modify logical preferences in the .lpf file using Diamond's preference-editing views, or you can modify the .lpf file directly using a text editor. Typically, you will start with a small .lpf file, which might contain basic timing preferences. For a complex design, the .lpf file might be modified several times during the course of design as timing and placement goals are refined.

The .lpf file can also be used to override attributes that exist in the HDL source. When you modify these constraints in the .lpf file, they will take precedence over those in the HDL file.

Unlike the .prf file, which is generated by the map process, the .lpf file contains only user-defined preferences.

Working with Multiple .lpf Files Diamond enables you to add new or existing preference files to your project and experiment with different sets of constraints. Each added .lpf file is listed in the LPF Constraint Files folder on Diamond's File List view. You can then activate a different .lpf file for a selected implementation.

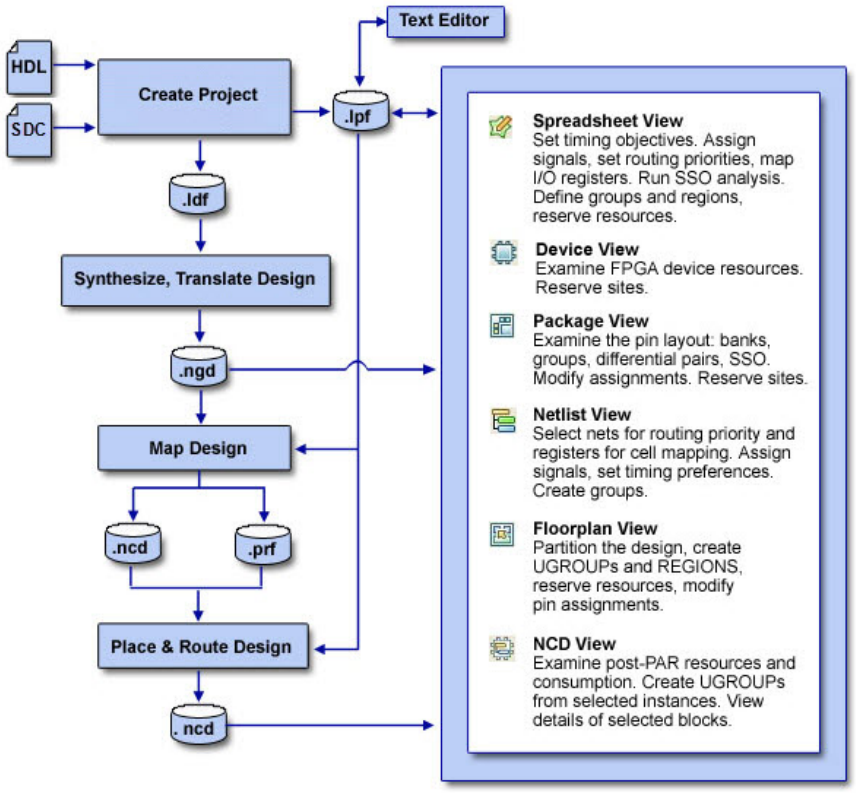
See Also
- "Adding a Logical Preference File to a Project" on page 459
- "Activating a Preference File" on page 461
- "Managing Project Sources" on page 34
- "Integrated Synthesis" on page 583
- "Preferences and the Design Flow" on page 377
- "Preference-Editing Views and Memory" on page 407
- "Physical View" on page 445
- "Logic Block View" on page 447

Preference Flow

Diamond provides six views for editing preferences. You can also modify preferences manually in the logical preference file (.lpf) using a text editor. After routing, you can examine finer details in the read-only Physical View and Logic Block View.

Synthesis constraint files (SDC) can be processed using either Lattice Synthesis Engine (LSE) or Synplify. Refer to "Multiple Entry Constraint Flow" on page 360.

Figure 50:



See Also ▶"Multiple Entry Constraint Flow" on page 360

- "Preferences and the Design Flow" on page 377
- "Preference-Editing Views and Memory" on page 407
- "Physical View" on page 445
- "Logic Block View" on page 447

Preferences and the Design Flow

Design constraints, including logical preferences and HDL attributes, work on design elements that are present in the logical domain, which is the user's netlist. In the design flow, logical preferences are specified in the logical preference file (.lpf), usually after synthesis and translation. HDL attributes are specified in the source files and are transferred into the binary native generic database (.ngd) file through logic synthesis and the Translate Design process

(ngdbuild). The mapping process maps logical .ngd elements to physical elements in the native circuit description file (.ncd), and it translates logical preferences to physical preferences in the physical preference file (.prf).

Logical Domain References The logical domain refers to the objects in the .ngd, which is a direct translation of the EDIF netlist. The following HDL example, “Verilog Hierarchical Design,” illustrates the objects that will be transferred to the .ngd by the synthesis and translation processes.

Figure 51:

```
module verilog_hierarchical_design(q, a, b, sel, r_l, clk,
rst);
output [7:0] q;
input [7:0] a, b;
input sel, r_l, clk, rst;
wire [7:0] mux_out, reg_out;

mux mux_1 (.out(mux_out), .a(a), .b(b), .sel(sel));

reg8 reg8_1 (.clk(clk), .data(mux_out), .q(reg_out),
.rst(rst));

rotate rotate_1 (q, reg_out, clk, r_l, rst);

endmodule
```

In the following .lpf file for the “Verilog Hierarchical Design” example, the timing, floorplan, and I/O preferences are defined in terms of objects of the logical domain—the ports, nets, or cell instances from the “verilog hierarchical design” cell in the EDIF file. The .lpf file illustrates package/floorplan locations of the FPGA device such as package locations P7, K4, and E13.

Figure 52:

```
COMMERCIAL ;
BLOCK RESETPATHS ;
BLOCK ASYNCPATHS ;
# Timing Preferences
FREQUENCY PORT "clk" 66 MHz;
INPUT_SETUP ALLPORTS 2.5ns CLKPORT "clk";
# Floorplan Preferences
UGROUP "Crit_Path"
  BLKNAME mux_1/out_4
  BLKNAME rotate_1/q_7
  BLKNAME reg8_1/q_1;
# I/O Preferences
LOCATE  COMP  "clk"  SITE  "P7";
LOCATE  COMP  "rst"  SITE  "K4";
LOCATE  COMP  "sel"  SITE  "E13";
```

While logic synthesis might rename or adjust the original design hierarchy, it is generally easy to relate logical domain references to the original HDL or schematic-based model of the design.

The Map Program The map program performs two major roles in the Diamond design flow:

- It maps the .ngd library elements into elements in the native circuit description file (.ncd), which is referred to as the physical domain. Logical elements of the .ngd (such as ports, flip-flops, and LUTs) are mapped to physical elements of the .ncd (such as slices and I/O components).
- It interprets all logical preferences from the .lpf and the .ngd files and, where appropriate, translates them into equivalent physical preferences in the physical preference file (.prf).

The map program also performs design rule checking when interpreting preferences. Logical constraints from the .lpf/.ngd are validated against the logical database (.ngd) file. Any logical references that do not exist in the .ngd are reported as semantic warnings and ignored by default. You can change the default by setting the “Ignore Preference Errors” property of the Map Design process to False in the project Strategy dialog box. This will cause the map program to error and will make these issues more visible.

When you update and save the .lpf during the design flow, the project gets initialized back to pre-map status and you must rerun the map process.

Note

If the time stamp of the .lpf for the project is newer than the post-map .ncd file, the Map Design and Place & Route Design processes will be reset. If you want to avoid resetting the project state, avoid saving any additions or changes to the .lpf file. You can still view the in-memory preferences in Preference Preview.

You can also use Timing Analysis View to experiment with sets of timing preferences and obtain quick analysis without rerunning Map and Place & Route. See “Using Timing Analysis View” on page 732.

Preferences in the .prf File The map process translates physical counterparts of logical preferences, as required, into the physical preference (.prf) file. Preferences listed in the .prf file result from both logical preferences and HDL attributes.

In the following .prf file for the “Verilog Hierarchical Design” example, the timing, I/O, and floorplan preferences of the .lpf have been translated by the design mapper to references that are compatible with the .ncd. In this example, most preferences appear unchanged; the object identifiers are common between the .ngd and .ncd. However, the UGROUP preference in the .lpf has been translated into a PGROUP with object references that reflect the post-map design and the consolidation of logical domain blocks into physical components.

Although it is possible to add preferences to the .prf file manually, this method is definitely not recommended because it would have to be done again each time after remapping.

Figure 53:

```
SCHEMATIC START ;
# map:  version Diamondv1.0_PROD_Build (133) -- WARNING: Map
write only section -- Fri Jun 11 09:06:59 2010

SYSCONFIG PERSISTENT=OFF CONFIG_MODE=SLAVE_SERIAL DONE_OD=ON
DONE_EX=OFF MCCLK_FREQ=2.5 CONFIG_SECURE=OFF WAKE_UP=21
COMPRESS_CONFIG=OFF INBUF=OFF ENABLE_NDR=OFF ;
PGROUP "Crit_Path"
  COMP "rotate_1/SLICE_4"
  COMP "reg8_1/SLICE_6"
  COMP "mux_1/SLICE_13";
LOCATE COMP "rst" SITE "K4" ;
LOCATE COMP "clk" SITE "P7" ;
LOCATE COMP "sel" SITE "E13" ;
FREQUENCY PORT "clk" 66.000000 MHz ;
SCHEMATIC END ;
BLOCK RESETPATHS ;
BLOCK ASYNCPATHS ;
INPUT_SETUP ALLPORTS 2.500000 ns CLKPORT "clk" ;
COMMERCIAL ;
```

Physical Domain References In cases where there is a one-to-one correlation between a logical object and a silicon device element, the physical domain reference uses the same identifier as the logical domain reference. This is common for objects such as ports, EBRs, PLL/DLLs, and DSP blocks. But objects that are consolidated into slices—such as ports, flip-flops and LUTs—are renamed with a SLICE identifier.

Floorplan View presents a graphical view of the physical placement of elements after the Place & Route process. The tool tips and status bar provide logic information and location for placed components. You can view the precise logical identifiers for a SLICE component by double-clicking the component on the Floorplan View layout to open Logic Block View. Logic Block View's schematic representation displays the logical names of registers within the slice.

You can also cross-probe to Netlist View to view the precise logical identifiers.

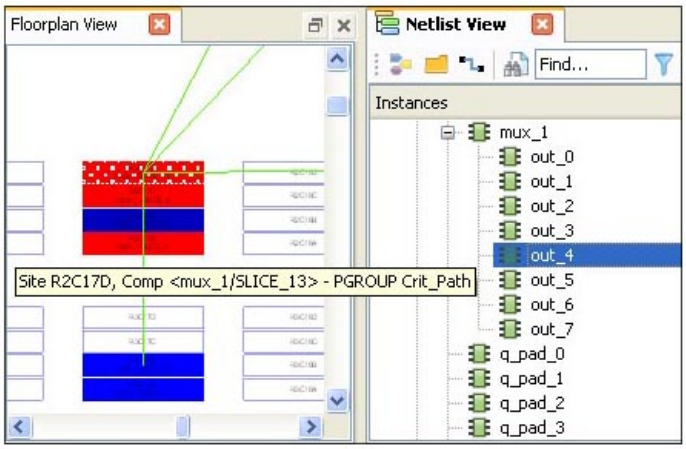
See Also ▶"HDL Attributes" on page 1282

- "The Logical Preference File" on page 375
- "The Physical Preference File" on page 405
- "Preference Conflict Resolution" on page 395

Preferences to Improve Timing

Timing preferences control the timed paths in an FPGA design. They are essential for good placement and routing results and for complete timing analysis. The following timing preferences are available in Spreadsheet View:

Figure 54:



Global Preferences Sheet – Block Path

- Block Asynchpaths – Blocks all paths from pads to FF inputs for the frequency and period preferences.
- Block Reset Paths – Blocks timing constraints on all asynchronous reset paths in the design.
- Block RD During WR Paths – Prevents timing analysis of RAM executing a read-during-write on the same address in a single clock period. This is recommended if your design contains PFU-based RAMs.

Timing Preferences Sheet

- FREQUENCY/PERIOD – Specify a frequency or period requirement on each clock net or clock port in the design. The PAR_ADJ keyword for PERIOD and FREQUENCY allows you to vary timing requirements for place-and-route results versus the requirements reported by TRACE, the timing analysis tool. This allows you to experiment more efficiently with over-constraining your design to achieve the best possible timing results.
- INPUT_SETUP – Each clocked input should have an input setup preference.
- CLOCK_TO_OUT – Each clocked output should have a clock-to-out preference.
- Block NET or PATH – All nets or paths specified should be ignored (blocked) by timing analysis.
- MULTICYCLE – Specify which paths' timing constraints can be relaxed.
- MAXDELAY – Identify a maximum total delay for a net, bus, or path in the design

After placement and routing, you can experiment with different sets of timing preferences using Timing Analysis View, which provides a timing preference

file (tpf) version of Spreadsheet View. Included in the TPF Spreadsheet View are Global, Timing, and Group preference sheets.

Note

The map report (.mrp) contains an “ASIC Components” section that lists the ASIC instances and type used in the design. All other registers in the report are CELLs. There might be times when you need to apply a timing preference, such as MULTICYCLE or MAXDELAY, that requires specification of CELL or ASIC; but you are not sure whether the specific instance is an ASIC or a CELL. In these situations, run the Map Report process.

Path Checking on Timing Preferences When a timing preference is specified, the software does a path check. The path checked depends upon the preference. The following table illustrates the path checks made for each timing preference:

Timing Preference	Description
PERIOD/FREQUENCY	On a clock net, generates a check for delays on all paths that terminate at a pin that has a setup or hold timing constraint relative to the clock net.
MAXDELAY PATHCLASS	Checks the delay of the specified class of paths: OUTPUTPATHS — all paths to primary output pins of the device CLOCKPATHS — all paths to clock inputs of SLICE/FSLICEs and PICs ENABLEPATHS — all paths to clock enable inputs of SLICE/FSLICEs SYNCPATHS — all paths to SLICE/FSLICE and PIC synchronous inputs ALLPATHS — a superset of these path classes
CLOCK_TO_OUT/ INPUT_SETUP	Specifies the delay offset for an external data input/ output relative to an external clock input, and generates delay checks on syncpaths and outputpaths.
BLOCK	Provides a means of blocking timing checks on specific nets, paths (containing the specified net), or path classes reported in the verbose report. RESETPATHS — specifies all asynchronous set/ reset paths, through an asynchronous set/reset pin on a design component. ASYNCPATHS — All paths from pads to FF inputs will be blocked for the frequency and period preference.
MULTICYCLE	Allows for relaxation of previously defined PERIOD or FREQUENCY constraints on a path.

See Also ▶“Auto Generating Timing Preferences” on page 492

- "Using Timing Preference Files (.tpf)" on page 505
- "Analyzing Static Timing" on page 694
- "Achieving Timing Closure" on page 571
- "Preferences" on page 1195
- "HDL Attributes" on page 1282

Preferences for Floorplanning

Floorplanning involves the placement of logical component instances or signals of a synthesized design to package pins or other sites of the device floorplan to improve the design's performance.

I/O Assignments I/O planning involves both the configuration of the programmable sysIO buffers to meet signal standard requirements as well as package pin locations to meet signal routing specification of the printed circuit board design.

The FPGA's PIO buffer varies by location around the chip die and might support a subset of potential signal standards, depending on the location. Some pins might have special dedicated connections to programming or clocking circuits of the chip and put restrictions on how pin assignments can be made. Diamond's Package View and Spreadsheet View enable you to define the correct preferences for pin assignments (LOCATE COMP), port groups (DEFINE GROUP) and port attributes (IOBUF).

See the sections "Assigning Signals" on page 466; "Setting Port, Net, and Cell Attributes" on page 475; and "Defining Port, Cell, and ASIC Groups" on page 506.

Logical Groups One of the main floorplanning methods is to group logical instances into a universal group (UGROUP) or hierarchical group (HGROUP). UGROUP is intended for grouping blocks in hierarchies or no hierarchy. HGROUP, which is usually defined as part of a VHDL architecture or a Verilog module declaration, is used to infer a unique placement group for each instance of a block. UGROUPs and HGROUPs can include PFU, PFF, EBR, and DSP-based logic.

UGROUPs and HGROUPs are translated by the design mapper (map) into PGROUPs, which directs the placer algorithm of PAR to place the group members in proximity. Diamond's Spreadsheet View, Netlist View, and Floorplan View enable you to create and edit UGROUPs

Note

The Diamond preference views do not support the creation or editing of HGROUPs.

UGROUPs and HGROUPs and their related attributes facilitate partitioning the logical design. When a UGROUP or HGROUP is defined, the mapping program prevents logical elements of different groups from being mapped into the same block and achieves a natural partitioning of the mapped design.

After mapping, the place and route process interprets the groups and places the grouped physical elements in proximity, effectively minimizing the connections between them.

See "Grouping Logical Components" on page 506.

Placement Groups Placement groups (PGROUP) are inferred by the Map process from any HGROUP and UGROUP definitions that exist in the logical preference (.lpf). When you use UGROUPs and HGROUPs, you are using the logical domain, which gives you control over how functions are grouped relative to each other. After mapping, these groups appear in the physical preference (.prf) file as PGROUPs.

Note

The legacy PGROUP preference is no longer supported for grouping PFF, PFU, EBR, and DSP-based logic in the logical preference file. Use the UGROUP preference instead.

The DEFINE GROUP preference should be used instead of the legacy PIO-based PGROUP preference. Although you can still create a PIO-based PGROUP in the logical preference file using a text editor, this is not recommended. The Diamond preference views do not support the display or editing of PIO-based PGROUPs.

See Also ▶"UGROUPs" on page 393

Preferences for Programming

The Global preferences sheet allows you to set configuration

The Lattice sysCONFIG feature is the non-JTAG interface supported by Lattice FPGA devices. The keywords for the sysCONFIG preference, which are available on the Global Preferences sheet, enable you to

Global Preferences

Global preferences are constraints that affect the entire design. They include preferences for setting voltage and temperature, preferences for blocking timing analysis on all paths of a certain type, configuration (sysCONFIG) preferences, and user code. Most of these preferences are available from the

Global sheet of Spreadsheet View. The list of available preferences varies by device family. See the references below the table for more information about sysCONFIG features for Lattice FPGA device families.

Table 19:

Preference Name	Supported Devices	Description
BACKGROUND_RECONFIG	ECP5U/UM, MachXO2, MachXO3D, MachXO3L	**NOTE:** The use of this preference is recommended for advanced users only. ON – Allows user to reconfigure device in user mode. OFF (default) – Disables this preference. SRAM_ONLY - specific for MachXO3D. SRAM_EBR - specific for MachXO3D.
BACKGROUND_RECONFIG_SECURITY	MachXO3D	ON - CFG can read/write to ESB. OFF (default) - CFG can read only.
BANK	All	Specifies the VCCIO voltage level of the IO bank.
BANDGAP	MachXO2	ON (default) – Statically enables analog PLL circuitry, power-on reset, oscillator, and referenced I/Os. OFF – Disables this preference.
BLOCK ASYNCHPATHS	All	ON (default) – Blocks timing analysis on all paths from pads to FF inputs for the FREQUENCY and PERIOD preference. OFF – Disables this preference.
BLOCK INTERCLOCK DOMAIN PATHS	All	**NOTE!** The use of this preference is risky and not recommended. ON – Specifies all paths involving data transfer between registers that are clocked by different clock nets—even when those clock nets are related. OFF (default) – Disables this preference.
BLOCK JITTER	All	ON – Blocks timing analysis on input and system jitter. OFF (default) – Disables this preference.
BLOCK RD DURING WR PATHS	All	ON – Globally blocks timing analysis on RAM read paths during the write operation. OFF (default) – Disables this preference.
BLOCK RESETPATHS	All	ON (default) – Blocks timing analysis on all asynchronous set/reset paths, through an asynchronous set/reset pin on a design component. OFF – Disables this preference.
COMPRESS_CONFIG	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	ON (default) – Generates a compressed version of the bitstream file. The device will route the compressed bitstream through the decompression engine during configuration. OFF – Disables this preference. It is possible for the compressed bitstream to be larger than the uncompressed bitstream.

Table 19:

Preference Name	Supported Devices	Description
CONFIG_IOVOLTAGE	ECP5, LatticeECP3	Sets the voltage for the sysCONFIG bank. This attribute tells the software the voltage that is required in this bank to satisfy the sysCONFIG requirements.
CONFIG_MODE	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeXP2, LatticeSC/M	Specifies the configuration mode. The configuration modes available, as well as the default mode, will depend on the targeted device.
CONFIG_SECURE	All	Allows you to set the security of the device. ON – Blocks Read Back of the configuration memory. OFF (default) – Enables Read Back through any port. **Note**: For LatticeECP2 devices, use the Security Settings interface from the Diamond Tools menu to set this preference.
CONFIGURATION	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	Specifies settings for storing the configuration bitstream. CFG (default) – Stores the configuration bitstream, including EBR init, in the configuration flash array and does not allow it to overflow into the UFM. CFG_EBRUFM – Stores the configuration bitstream (not including EBR init) in the configuration Flash array and stores EBR init data in the lowest page addresses of the user flash memory (UFM). CFGUFM – Stores the configuration bitstream, including EBR init, in the configuration flash array and allows it to overflow into the UFM. EXTERNAL – Stores the configuration bitstream, including EBR init, in an external memory device.
CUR_DESIGN_BOOT_LOCATION	MachXO3D	Determines the.jed file downloaded to Flash memory: IMAGE_0 (default) - <project_name>_a.jed is generated. IMAGE_1 - <project_name>_b.jed is generated.
CUSTOM_IDCODE	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	Specifies a 32-bit value to be used as the JTAG ID of the device when the MY_ASSP feature is enabled.
CUSTOM_IDCODE_FORMAT	MachXO3D	Specifies the CUSTOM_IDCODE format. The available formats are HEX and BINARY (default).
DONE_EX	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeXP2, LatticeSC/M	Determines if the wake-up sequence is triggered by an external DONE signal. ON –- Delays wake-up until the DONE pin is driven high by an external signal and synchronous to the clock. OFF (default) – Synchronously wakes up when the internal Done bit is set and ignore any external driving of the DONE pin. If DONE_EX is set to ON, DONE_OD should be set to the default value of ON.

Table 19:

Preference Name	Supported Devices	Description
DONE_OD	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeXP2	ON (default) – Configures the DONE pin as an open drain configuration. OFF – Disables this preference.
DONE_PULL	ECP5, LatticeSC/M	ON (default) – Configures the DONE pin with a pull up. OFF – Disables this preference.
DUALBOOTGOLDEN	MachXO3LF	INTERNAL (default) – Controls the Dual Boot Golden File Location. This setting specifies an internal location. EXTERNAL – Specifies an external Dual Boot Golden File Location.
ENABLE_NDR	LatticeECP2/M, LatticeECP3	OFF (default) – I/Os are tri-stated during reconfiguration. ON – Enables Non-disruptive Reconfiguration (NDR), which allows the device to be reconfigured while I/Os are in Leave-Alone mode.
ENABLE_TRANSFR	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	DISABLE (default) – Leaves the Freeze_IO and Freeze_MIB fuses off and disables the TransFr function. ENABLE – Turns on the Freeze_IO and Freeze_MIB fuses to enable the TransFR function.
GSR_NET	All	Assigns a specified net as the input to the global set/reset (GSR) buffer inferred by the design mapper (MAP).
I2C ADDRESS	All	I2C ADDRESS SIZE - Specify 7 or 10 bit. I2C ADDRESS - Specify the starting address
I2C_PORT	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	ENABLE – Makes the I^2C port active for configuration access by preventing the dual-purpose pins from being used as I/O pins. DISABLE (default) – Makes the dual-purpose pins related to the I^2C port available as user I/O pins.
INBUF	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeXP, LatticeXP2, MachXO	On – Turns on the input buffers. You must set this preference to On when you need to perform boundary scan testing. Off – Disables all unused input buffers to save power. ECP5, LatticeECP/EC, LatticeECP2/M: defaults to Off LatticeXP, LatticeXP2, MachXO,: defaults to On
JTAG_PORT	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	ENABLE (default) –- Prevents the JTAG pins from being used as ordinary I/Os. DISABLE – Allows the assigning of signals to the JTAG pins. A high signal on the JTAGENB would then be required to re-enable the JTAG port, after which the 4 pins would become dedicated JTAG pins.

Table 19:

Preference Name	Supported Devices	Description
MASTER_PREAMBLE_DETECTION_RETRY	MachXO3D	Sets the number of times the preamble detection should be retried in master mode. The attribute value is 0 to 3. 0 (default) - no retry 1 - try one time 2 - try two times 3 - try three times
MASTER_PREAMBLE_DETECTION_TIMER	MachXO3D	Sets the detection timer count value in master mode. The attribute value is 0 to 15. 0 (default) - approx. 126Kus 1 - 63 Kus, ... 13 - 15.4 us 14 - 7.7 us 15 - 3.85us
MASTER_SPI_PORT	ECP5, LatticeXP2, MachXO2, MachXO3D, MachXO3L, Platform Manager 2 except MachXO3D	ENABLE – Allows SRAM to be configured using the external Master SPI port. DISABLE (default) – Prohibits the use of the external Master SPI port for SRAM configuration. EFB_USER (MachXO2, MachXO3D, and MachXO3L only) – Reserves the SPI pins for user Master SPI mode in the embedded function block (EFB) In MachXO3D, MASTER_SPI_PORT is not open for user setting.
MCCLK_FREQ	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeSC/M, MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	When you have determined that the a device will be a Master Configuration device and will provide the clocking source for configuration, the CCLK will become an output clock with the frequency set by the user. At the start of configuration the device operates with the default Master Clock Frequency of 2.5MHz. One of the first configuration bits set will be the Master Clock. Once the Master Clock configuration bits are set, the clock will start operating at the user-defined frequency. MCCLK_FREQ is used to control the Master Clock frequency. It defaults to the lowest frequency depending on target device.
MUX_CONFIGURATION_PORTS	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	ENABLE – Controls dynamic muxing of JTAG and IOs with JTAGEN. DISABLE (default) – Prevents the disabling of all configuration ports.

Table 19:

Preference Name	Supported Devices	Description
MY_ASSP	LatticeXP2, MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	ON – For LatticeXP2 devices, causes the USERCODE to be used as a custom device ID. For MachXO2, MachXO3D, and MachXO3L devices, changes the JTAG IDCode of the device to the contents of the CUSTOM_IDCODE field. OFF (default) - Uses the regular Lattice device ID code.
ONE_TIME_PROGRAM	LatticeXP2, MachXO2, MachXO3D, MachXO3L, Platform Manager 2 except MachXO3D	ON – Protects the Flash memory from erasure or reprogramming. OFF (default) – Allows erasure and programming of the Flash memory.
PERSISTENT	LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeSC/M	When using the sysCONFIG port, this keyword must be set to ON (except for LatticeECP3 devices) to reserve the dual-purpose pins for configuration. The ON option lets the software know that all the dual purpose configuration pins will NOT be available for the fitter to use. For LatticeECP3 devices, the SLAVE_PARALLEL option reserves all sysCONFIG pins; and the SSPI option reserves only the slave SPI port.
POR	MachXO2	ON (default) – Statically enables power-on reset when BANDGAP is turned on. OFF – Disables this preference.
PRIMARY_BOOT	MachXO3D	Sets the silicon boot option. IMAGE_0 (default) - for boot from CFG0/UFM0 IMAGE_1 - for boot from CFG1/UFM1 EXTERNAL - for external boot LATEST - during the dual internal boot, selects the last loaded bitsteam first FORMER - during the dual internal boot, selects the first loaded bitstream
	MachXO3D	Enable enforcement of roll back control. ENABLE/DISABLE (default)

Table 19:

Preference Name	Supported Devices	Description
SDM_PORT	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	Self-download mode port. DISABLE (default) – Permits the SDM pins to be used as regular I/O pins. PROGRAMN – Retains the PROGRAMN pin as a dedicated SDM pin. PROGRAMN_DONE – Retains the PROGRAMN and DONE pins as dedicated SDM pins. PROGRAMN_DONE_INITN – Retains the PROGRAMN, DONE, and INITN pins as dedicated SDM pins. DONE - Retains the DONE pin as dedicated SDM pin. INITN - Retains the INITN pin as dedicated SDM pin.
SECONDARY_BOOT	MachXO3D	Sets the silicon boot option. NONE (default) - for single boot IMAGE_0 - for boot from CFG0/UFM0 IMAGE_1 - for boot from CFG1/UFM1 EXTERNAL - for external boot LATEST - during the dual internal boot, selects the last loaded bitsteam first FORMER - during the dual internal boot, selects the first loaded bitstream
SFDP_CHECK	MachXO3D	Check for Serial Flash Discoverable Parameter. DISABLE (default) - no SFDP check ENABLE_SFDP - SFDP checks and stops when SFDP fails ENABLE_SFDP_PREAMBLE - SFDP checks and continues PREAMBLE when SFDP fails.
SHAREDEBRINIT	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	ENABLE – Allows sharing of the same init file used by multiple EBRs. DISABLE (default) – Does not allow sharing of the init file.

Table 19:

Preference Name	Supported Devices	Description
SLAVE_IDLE_TIMER	MachXO3D	Sets the Slave Idle Timer count value. 0 (default) - Infinite (disabled) 1 - 128,000 Ms 2 - 64,000 Ms 3 - 32,000 Ms 4 - 16,000 Ms 5 - 8,000 Ms 6 - 4,000 Ms 7 - 2,000 Ms 8 - 1,000 Ms 9 - 640 Ms 10 - 320 Ms 11 - 160 Ms 12 - 80 Ms 13 - 40 Ms 14 - 20 Ms 15 - 10 Ms
SLAVE_PARALLEL_PORT	ECP5	ENABLE – Makes the 8-bit slave parallel port available after configuration. DISABLE (default) – Prohibits the 8-bit slave parallel port from accessing the SRAM after configuration and allows the dual purpose pins to be used as general purpose IO's.
SLAVE_SERIAL_PORT	LatticeECP4U/M	ENABLE – Makes the slave serial port available after configuration. DISABLE (default) – Prohibits the slave serial port from accessing the SRAM after configuration, and allows the dual purpose pins to be used as general purpose IO's.
SLAVE_SPI_PORT	ECP5, LatticeXP2, MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	ENABLE (default) – Allows on-chip Flash to be programmed and read back using the external Slave SPI port. DISABLE – Prohibits the use of the external Slave SPI port for on-chip Flash programming.
SPIM_ADDRESS_32BIT	MachXO3D	Extends the SPIM address. DISABLE (default) - 24-bit SPIM address ENABLE - 32-bit SPIM address
STRTUP	LatticeECP3	EXTERNAL (default) | TCLK | CCLK | MCLK Specifies the clock source to be used for synchronization with the start-up sequence.

Table 19:

Preference Name	Supported Devices	Description
SYSTEM_JITTER	All	Specifies the peak-to-peak system jitter for timing analysis. The value specified overwrites the default value. It should not be smaller than the default value for the device.
TEMPERATURE	All	Assigns an operating junction temperature for the part that changes the derating factor for timing.
TRACEID	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	Specifies a code for storing device data, much like USERCODE but in binary format.
TRANSFR	ECP5	OFF (default) – Leaves the Freeze_IO and Freeze_MIB fuses off and disables the TransFr function. ON – Turns on the Freeze_IO and Freeze_MIB fuses to enable the TransFR function.
UNIQUE_ID	ECP5	Defines the upper sixteen bits of an automatic 32-bit USERCODE that will be used to identify the design.
USERCODE	All	Enter binary, hex, or ASCII values to program the Usercode bits. **Note**: The use of CHECKSUM as USERCODE is supported for LatticeXP2 devices. AUTO format, which enables UNIQUE_ID, is supported for ECP5 devices.
USERCODE_FORMAT	All	Specifies selected format of USERCODE. The available formats are Hex, Binary (default), and ASCII.
VOLTAGE	All	Assigns a value for nominal FPGA core voltage
WAKE_ON_LOCK	LatticeXP2, LatticeECP3	Determines whether the device will wait for the PLL to lock before beginning the wake-up process. OFF (default) – the device will wake up regardless of the state of the PLL lock signal. ON – the device will not wake up until the PLL lock signal for the given PLL is active.
WAKE_UP	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeXP2, LatticeSC/M	Specifies the wake-up sequence. The default value is determined by the DONE_EX setting. DONE_EX = OFF: defaults to 21 and ranges between 1 and 25 DONE_EX = ON: defaults to 4 and ranges between 1 and 7

See Also ▶Setting Global Preferences

- TN1260, *ECP5 sysCONFIG Usage Guide*
- TN1053, *LatticeEC/ECP sysCONFIG Usage Guide*
- TN1108, *LatticeECP2/M sysCONFIG Usage Guide*

- TN1169, *LatticeECP3 sysCONFIG Usage Guide*
- TN1080, *LatticeSC sysCONFIG Usage Guide*
- TN1082, *LatticeXP sysCONFIG Usage Guide*
- TN1141, *LatticeXP2 sysCONFIG Usage Guide*
- TN1204, *MachXO2 Programming and Configuration Usage Guide*
- TN1279, *MachXO3L Programming and Configuration Usage Guide*
- TN1303, *CrossLink Programming and Configuration Usage Guide*
- FPGA-TN-02069 - *Programming and Config Usage Guide*

UGROUPs

A universal group (UGROUP) is a group of logical components that are to be packed close together. A UGROUP instructs the mapper not to pack unrelated logic into the same block. The UGROUP will be translated by the mapping process to a placement group (PGROUP) in the physical preference file (.prf), giving instructions to the placer to place the components in close proximity to each other. These groups are for mapping and placement only; they have no direct timing influence. See "DEFINE GROUP" on page 1213 for a different type of group that is usable for timing constraints.

Note

The Diamond preference views do not support the creation or editing of PGROUPs.

UGROUPs can be defined in the logical preference file (.lpf) or in the HDL source.

UGROUPs can include anchored or floating PFU, PFF, and EBR blocks, which can be bounded or unbounded. UGROUPs that include DSP blocks must be anchored. If a floating UGROUP includes a DSP block in the .lpf file, the system will issue a warning and ignore the block during mapping, placement and routing.

UGROUPs in the Logical Preference File

You can create or edit UGROUPs manually with a text editor using the ASCII .lpf file and the UGROUP preference syntax. You can also create UGROUPs using the Diamond preference views. UGROUPs from the HDL source are displayed in Spreadsheet View. If the UGROUPs are anchored (bounded or unbounded), they will also be displayed in Floorplan View. After being modified and saved, they will appear in the .lpf file. Use any of the following methods to create or edit UGROUPs:

- In the Group sheet of Spreadsheet View, create a new UGROUP and add instances; or modify an existing UGROUP.
- In Netlist View, create a new UGROUP from selected instances.
- In NCD View, create a new UGROUP from selected instances.

- In Floorplan View, draw a new UGROUP bounding box and add instances; modify an existing UGROUP; or create a new UGROUP from selected routed instances.

When you issue the “Save” command from any preference view, Diamond writes the new and modified UGROUPs to the currently active .lpf file.

UGROUPs in the HDL

Two main logical group preferences are available as attributes in the HDL and are displayed in the Group sheet of Spreadsheet View after the Translate Design process. They include:

UGROUP – The UGROUP attribute is used in the HDL for grouping components within hierarchies or across hierarchies. When you run the Translate Design process and open the design in Spreadsheet View, you will see these groups listed in the Group preference sheet. If you modify and save these groups, they will also appear in the .lpf file.

HGROUP – The UGROUP attribute is almost always the type of logical group that is used. However, the HGROUP attribute (also referred to as hierarchical group), can be used for grouping components within a module that is to be instantiated multiple times. During synthesis, the HGROUP is expanded into individual instances. Each instance is then displayed as a unique UGROUP in the Group sheet of Spreadsheet View. After you modify these groups and save the design, they will appear in the .lpf file. HGROUPS are not commonly used, and their functionality can be duplicated by assigning UGROUPs with a distinct name to each instance.

Note

The Diamond preference views do not support the creation or editing of HGROUPs.

Logical groups specified in the HDL design file are displayed in the Group sheet of Spreadsheet View. However, logical groups that are created in the Spreadsheet View are kept in the post-synthesized logical preference file (.lpf) and are not back annotated to the HDL design file.

Note

HDL groups that are modified and saved in Diamond are added to the .lpf file. Afterwards, the .lpf file takes precedence. If you want the original HDL preference to be honored, you must delete the conflicting preference in Diamond or you must delete it directly from the .lpf file using a text editor.

See Also ▶“Preferences for Floorplanning” on page 383

- “Creating UGROUPs” on page 509

Preference Conflict Resolution

When more than one preference is ascribed to a single design element, some preferences will take precedence over others, depending on the level of specificity, the order of the conflicting preferences, and redundancy between the HDL source and the logical preference file (.lpf). The software resolves such conflicts by adhering to the following rules of precedence:

- Timing preferences that have a higher priority than others take precedence when the same design objects are constrained in two or more preferences. For example, the higher-priority BLOCK preference takes precedence over the lower-priority MULTICYCLE preference.

  See "Timing Preference Priority" on page 395

- When more than one preference applies to a net or path, more specific preferences are honored before less specific ones. For example, individual net/path preferences supersede group (bus) preferences, and group preferences supersede global preferences.
- When more than one preference at the same specificity level exists for a block/comp/net/path, the preference that occurs last in the preference file takes precedence.

  Note

  MAXDELAY PATH (FROM/TO) preferences will supersede ALLPATHS, PERIOD, or FREQUENCY preferences where possible.

- When an attribute in the HDL conflicts with a preference in the logical preference file, the preference in the .lpf file has precedence. For example, when a given register is included in both a UGROUP attribute and a logical preference definition, the logical preference is honored.

  See "Conflicting HDL Attributes, Constraints, and Preferences" on page 401 and "Conflicting IOBUF Constraints" on page 398

- HGROUPs and UGROUPs in the HDL do not appear in the .lpf unless they have been modified and saved to the .lpf file. Afterwards, the groups in the .lpf file take precedence over those in the HDL. Conversely, logical groups created in the Diamond preference views are kept in the .lpf file and are not back annotated to the HDL design file.
- When two or more UGROUPs are identified with the same name in the .lpf file, the group that appears last is honored.
- When different UGROUPs in the .lpf file contain one or more of the same members, the UGROUP that appears last is honored.

Timing Preference Priority

The following list shows the priority of timing preferences, beginning with the highest priority BLOCK preference and ending with the lowest priority PERIOD/FREQUENCY preference:

1. BLOCK
2. INPUT_SETUP/CLOCK_TO_OUT

3. MULTICYCLE
4. MAXDELAY
5. PERIOD/FREQUENCY

In the following example, two preferences are assigned to port Q.

Figure 55:

```
MAXDELAY TO PORT "Q" 10 ns;
BLOCK PATH TO PORT "Q";
```

The BLOCK PATH TO PORT “Q” preference negates the MAXDELAY preference. The resulting TRACE Report (.twr) file lists both preferences:

Figure 56:

```
================================================================
   Preference: MAXDELAY TO PORT "Q" 10.000000 ns ;
               0 items scored, 0 timing errors detected.

  --------------------------------------------------------------------------

============================================================

   Preference: BLOCK PATH TO PORT "Q" ;
    1 item scored, 0 timing errors detected.
  ------------------------------------------------------------
```

The TRACE report does not indicate that the MAXDELAY preference has passed the constraint requirements, and it was not scored. If you remove the BLOCK preference and retain just the MAXDELAY preference, the TRACE report will show that the preference passes the constraint of 10 ns.

For timing preferences with the same priority level, more specific preferences are honored over less specific ones. For example:

In this example, net W gets 10 ns because this preference is more specific than ALLNETS

Redundant Timing and I/O Assignments In the following timing preferences example, order and specificity determine the preference that will be honored.

In this example, the MAXDELAY priority is as follows:

- Net W gets 15 ns because this preference is more specific than BUS or ALLNETS and it comes after the 10 NS preference.
- Net X gets 25 ns because the BUS A preference is more specific than ALLNETS.
- Net Y gets 25 ns because the BUS A preference comes after the BUS B preference.

Figure 57:

```
================================================================
   Preference: MAXDELAY TO PORT "Q" 10.000000 ns ;
               1 item scored, 0 timing errors detected.
   ----------------------------------------------------------

Passed:  The following path meets requirements by 7.415ns

Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

Source: FF         Q              Q_0io   (from CLK_c +)
      Destination:    Port       Pad            Q

Delay: 2.585ns  (66.8% logic, 33.2% route), 2 logic levels.

    Constraint Details:
2.585ns physical path delay d_MGIOL to Q meets
         10.000ns delay constraint by 7.415ns

    Physical Path Details:

Name    Fanout   Delay (ns)          Site               Resource
   C2OUT_DEL   ---     0.384   IOL_T24A.CLK to  IOL_T24A.INFF d_MGIOL (from CLK_c)
   ROUTE         1     0.858  IOL_T24A.INFF to      B14.PADDO Q_c
   DOPAD_DEL   ---     1.343      B14.PADDO to        B14.PAD Q
                      --------
                       2.585   (66.8% logic, 33.2% route), 2 logic levels.

  Report:    2.585ns is the maximum delay for this preference.
```

Figure 58:

```
MAXDELAY NET W 10 NS;
MAXDELAY ALLNETS 30 NS;
```

Figure 59:

```
MAXDELAY NET W 10 NS;
MAXDELAY ALLNETS 30 NS;
DEFINE BUS B NET Y NET Z;
DEFINE BUS A NET Y NET X NET W;
MAXDELAY BUS B 20 NS;
MAXDELAY BUS A 25 NS;
MAXDELAY NET W 15 NS;
```

- Net Z gets 20 ns because the BUS preference is more specific than ALLNETS.
- All other nets get 30 ns.

Note

MAXDELAY PATH (FROM/TO) preferences will supersede ALLPATHS, PERIOD, or FREQUENCY preferences if possible.

You can relax a global delay constraint with a more specific constraint under most conditions. For example, assume that there is a period preference of 50 ns on CLKB, which dictates that the maximum delay from STARTPOINT to ENDPOINT be 50 ns. You want to relax the period from STARTPOINT to ENDPOINT to 100 ns. To do this, you must define the STARTPOINT and the ENDPOINT, and then you must use the MAXDELAY FROM/TO preference.

Conflicting IOBUF Constraints

IOBUF constraints can sometimes conflict with one another, especially when they appear in both the HDL and the .lpf file or when they are used for individual ports as well as port groups. When there are conflicting IOBUF constraints, the following rules of precedence apply.

Attributes in the HDL and LPF for the Same Port or Port Group When conflicting IOBUF attributes exist in the .lpf file and in the HDL for the same port or port group, those in the .lpf file will take precedence.

Beginning with Diamond 3.2, IOBUF attribute changes that you make in Spreadsheet View for a specific port or port group will be combined with non-conflicting attributes that exist in the HDL for the same port or port group. When you save the changes, the combination will be recorded in the .lpf file.

For example, the HDL source assigns the following to Port A:

input A /* synthesis IO_TYPE=LVCMOS18 PULLMODE=UP LOC=12 */;

You open Spreadsheet View's Port Assignments sheet and change IO_TYPE to LVCMOS33 and set CLAMP to ON for Port A. When you save the changes, Diamond combines the non-conflicting HDL attributes with the Spreadsheet View changes and records them in the .lpf file as follows:

```
IOBUF PORT "A" CLAMP=ON IO_TYPE=LVCMOS33 PULLMODE=UP ;
```

The non-conflicting PULLMODE attribute from the HDL is added to the .lpf file, as is the CLAMP attribute that was set in Spreadsheet View. The LVCMOS18 IO_TYPE from the HDL is replaced by LVCMOS33 from Spreadsheet View.

The LOC=12 pin assignment from the HDL does not appear in the .lpf file, because it is a placement constraint. If you had changed the pin assignment to pin 14 in Spreadsheet View and saved it, it would appear in the .lpf file as a separate preference: `LOCATE COMP "A" SITE "14" ;`

When you set new IOBUF preferences manually using a text editor, conflicting IOBUF attributes in the .lpf file will completely override the HDL attributes for the same port or port group when the design is mapped. The exceptions to this rule are IO_TYPE and the placement constraint LOC.

- An IO_TYPE attribute that exists in the HDL will be honored unless the .lpf file contains an IO_TYPE constraint for the same port or port group.

- Beginning with Diamond 3.1, the placement constraint LOC does not get overridden by an IOBUF preference for the same port or port group unless it is redefined by a LOCATE preference in the .lpf file. See "Migrating Designs from Previous Releases" on page 400.

ALLPORTS Attribute and HDL Attributes All IOBUF attributes that exist in the HDL will get overridden by the Map process when the ALLPORTS attribute is used in the .lpf file. The one exception is IO_TYPE, which will be maintained unless the ALLPORTS attribute specifies a different IO_TYPE.

For example, the HDL source assigns PULLMODE=UP for Port A. Later, in Spreadsheet View, the IO_TYPE in the All Ports row is changed from LVCMOS25 to LVCMOS33. When the design is placed and routed, all of the ports are given the LVCMOS33 IO_TYPE, including Port A. But the PULLMODE for Port A reverts to DOWN, which is the default setting. The only way to return to the original PULLMODE=UP for Port A is to change it in the .lpf file and rerun Map and Place & Route.

When you use ALLPORTS in the .lpf file to make global IOBUF assignments, make sure that you move any existing HDL IOBUF assignments to the .lpf file.

Conflicting LPF Attributes for the Same Port or Group When two IOBUF preferences appear in the .lpf file for the same port or group, the one that occurs last will be honored and will override the one that appears earlier. The one exception is IO_TYPE, which be maintained unless the preference that occurs later specifies a different IO_TYPE. See "Combining Multiple IOBUF Preferences" on page 400.

For example, the following preferences for Port A appear in two different places in the .lpf file:

```
IOBUF PORT "A" SLEWRATE=SLOW PULLMODE=DOWN;


IOBUF PORT "A" SLEWRATE=FAST;
```

When the design is mapped, the SLEWRATE=FAST preference, which appears last, will be honored. The earlier preference will be completely overridden, which means that PULLMODE will revert to the default setting.

Conflicting LPF Attributes Between Ports, Groups, and ALLPORTS When the .lpf file contains IOBUF preferences that conflict among one or more ports, groups, and/or ALLPORTS, the more specific IOBUF preference takes precedence and will override the less specific preference. The one exception is IO_TYPE, which will be maintained unless the more specific preference specifies a different IO_TYPE.

For example, the following preferences show conflicting IOBUF attributes between Port A and a port group that includes Port A.

In the above example, a wildcard (A*) has been used to select ports for the "iogroup" port group, including Port A. When the design is mapped, the

```
DEFINE PORT GROUP "iogroup" "A*";
IOBUF GROUP "iogroup" IO_TYPE=LVCMOS33 PULLMODE=UP;
IOBUF PORT "A" IO_TYPE=LVCMOS18 CLAMP=OFF;
```

attributes for the more specific IOBUF Port A preference will be honored: IO_TYPE=LVCMOS18 and CLAMP=OFF.

In the following example, there is a conflict between the IOBUF preference for ALLPORTS and the IOBUF preference for Port A.

```
IOBUF ALLPORTS PULLMODE=UP;
IOBUF PORT "A" SLEWRATE=SLOW;
```

When the design is mapped, the more specific IOBUF preference for Port A will be honored. SLEWRATE will be set to SLOW for Port A, and the PULLMODE setting will revert to the default setting.

In the following example, there is a conflict between the IOBUF preference for ALLPORTS and the IOBUF preference for the “iogroup.”

```
IOBUF ALLPORTS IO_TYPE=LVCMOS33 CLAMP=ON;
IOBUF GROUP "iogroup" PULLMODE=UP;
```

When the design is mapped, only the IO_TYPE will be maintained from the ALLPORTS preference. PULLMODE will be set to UP for all ports in the “iogroup,” and CLAMP will revert to the default setting.

Migrating Designs from Previous Releases A design that was created with an earlier Diamond version might be affected by the following precedence rule change in Diamond 3.1:

- An IO_TYPE attribute in the .lpf file is no longer honored when there is a subsequent IOBUF preference in the .lpf file for the same port or port group. The IO_TYPE from the preference that appears first will either revert to the default or to one specified in the preference that appears last.
- A LOC attribute in the HDL will be honored. It will not get overridden by an IOBUF preference as long as there is no LOCATE preference for the same port or port group in the .lpf file.
- A BANK attribute in the HDL will be honored. It will not get overridden by an IOBUF preference as long as there is no IOBUF BANK preference for the same port or port group in the .lpf file.

When you use a design that was created in an earlier version, make sure that you move constraints from the HDL into the LPF file for the IOBUF attributes that you want to maintain.

Combining Multiple IOBUF Preferences When you have multiple IOBUF preferences in the .lpf file for the same port or port group, make sure that you combine them into one preference.

For example, the following two preferences appear in the .lpf file:

```
IOBUF PORT "a" IO_TYPE=LVTTL33 PCICLAMP=ON;

IOBUF PORT "a" PULLMODE=DOWN;
```

Except for the IO_TYPE, the second of these preferences, which appears last in the .lpf file, will override the first preference. This means that the PCICLAMP=ON will revert to the default OFF setting when the design is mapped. To maintain the PCICLAMP=ON setting, simply combine these two preferences into one.

```
IOBUF PORT "a" IO_TYPE=LVTTL33 PULLMODE=DOWN PCICLAMP=ON;
```

See Also
- "Conflicting HDL Attributes, Constraints, and Preferences" on page 401
- "Defining I/O Defaults" on page 475
- "Defining Port, Cell, and ASIC Groups" on page 506

Conflicting HDL Attributes, Constraints, and Preferences

Using a combination of HDL attributes and preferences is a powerful technique for constraining the implementation tools. However, it is possible to introduce a combination of HDL attributes and logical preferences that are redundant and in conflict with each other. When such conflicts occur, the preference in the .lpf file takes precedence.

Conflicting Pin Assignments: When there is a conflicting pin assignment between the HDL and the preference file, the assignment in the preference file will take precedence. For example, SPI_PIN_D has a LOC attribute of W25 in the source file. Later, in Spreadsheet View, the pin is assigned to site U21 and saved to the .lpf file. When the design is placed and routed, the U21 location is honored instead of the W25 assignment. The only way to return to the original W25 site assignment is to delete the LOCATE preference from the .lpf file.

The following is an HDL example of conflicting pin assignments:

Input SPI_PIN_D /* synthesis LOC = "W25" */;

The following is an LPF example of conflicting pin assignments, which overrites HDL:

LOCATE COMP "SPI_PIN_D" U21; (this is LPF, it over-rides HDL))

The same is true for IOBUF attributes. For example, the HDL assigns “my_signal” to L5 with OPENDRAIN set to ON. Later, OPENDRAIN is changed to OFF in Spreadsheet View. The pin assignment remains at L5, and the OPENDRAIN setting of OFF is honored.

See “Conflicting IOBUF Constraints” on page 398

Redundant UGROUP Declarations In the following example, two group preferences with the same name contain different members:

Figure 60:

```
UGROUP "Group_0"
    BLKNAME reg0
    BLKNAME reg1;
UGROUP "Group_0"
    BLKNAME reg2
    BLKNAME reg3;
```

When the design is mapped, the first occurrence of Group_0, with members reg0 and reg1, is ignored. Warnings will appear in the output log and in the map report.

The following example contains two group preferences with different names that contain the same members.

Figure 61:

```
UGROUP "Group_1"
    BLKNAME reg6
    BLKNAME reg7;
UGROUP "Group_2"
    BLKNAME reg6
    BLKNAME reg7;
```

When the design is mapped, Group_1 is ignored because its members conflict with the membership declared in Group_2 that appears second. Warnings will appear in the output log and in the map report.

Conflict between UGROUP and LOCATE COMP The following example contains a combination of group and location preferences that conflict:

Figure 62:

```
UGROUP "Group_0" BBOX 9 9
  BLKNAME I1
  BLKNAME I2;
LOCATE UGROUP "Group_0" SITE "EBR_R36C6" ;
LOCATE COMP "I1/pdp_ram_0" SITE "EBR_R36C22" ;
```

The Group_0 contains two blocks of the design. One element of block I1, an embedded block RAM (EBR) named pdp_ram_0, is placed at a specific EBR device site (EBR_R3C22) using the LOCATE preference.

Because LOCATE is more specific than a UGROUP preference, when the place and route program encounters this condition, it honors the LOCATE preference. It does not necessarily place pdp_ram_0 in proximity to the other members of Group_0. No warnings or errors are generated.

The following example contains a member (ip_ram_dq_0_1_0) that has a LOCATE preference. This results in the removal of this member from the UGROUP during placement. The LOCATE preference for this member will not influence the placement of the other members of the UGROUP, because this member has been removed from the UGROUP.

Figure 63:

```
UGROUP "grp1"
        BLKNAME ip_ram_dq_0_0_1
        BLKNAME ip_ram_dq_0_1_0;
LOCATE UGROUP "grp1" REGION "region_1" ;
LOCATE COMP "ip_ram_dq_0_1_0" SITE "EBR_R29C8" ;
```

The UGROUP “grp1” will continue to be put in “region_1”, but grp_1 now only has the single member: ip_ram_dq_0_0_1. The element ip_ram_dq_0_1_0 will be placed at site “EBR_R29C8”.

Redundant DEFINE GROUP Declarations Resolution of conflicting DEFINE GROUP preferences, such as port groups, is not handled consistently across all programs. The style of declaration shown in the following example should be avoided. It contains port groups that have the same name but contain different members.

Figure 64:

```
DEFINE PORT GROUP "Group_0"
"d0(0)"
"d0(1)" ;
DEFINE PORT GROUP "Group_0"
"d0(2)"
"d0(3)" ;
INPUT_SETUP GROUP "Group_0"INPUT_DELAY 5.000000 ns HOLD
3.000000 ns CLKPORT "CLK" ;
```

The first occurrence of Group_0 with members d0(0) and d0(1) is ignored by the preference views, however, the second occurrence of Group_0 with member d0(2) and d0(3) is ignored by the TRACE (trce) and PAR programs. Both groups are passed into the physical preference file by map. Ignored groups are not reported by the system in this example.

See Also ▶“The Logical Preference File” on page 375

▶ “The Physical Preference File” on page 405

Changes to Net Names Between Preferences and Reports

If you define certain preferences, such as a net name, and you re-run synthesis, the name of that net may be changed. If you have an opportunity to refer to a port versus a net, you should always use the port, because the name will be more constant from run to run.

If must specify a net, it's possible that the net name can change, so you must manage them.

One way specify net names is to specify those constraints through SDC. If a net constraint is specified in SDC, that constraint will get updated automatically by synthesis, because synthesis will write the logical preference file (.lpf) for each run. Refer to "Using SDC Constraints" on page 364.

In some instances, constrained net names in the logical preference file (.lpf) differ from those in the physical preference (.prf) file because optimization takes place during the mapping process. In most instances, the mapper honors the preference, althought it may change the net names in the .prf.

A message will be written to the output log when this happens to show each net name as it appears in the .lpf file and the conversion to the new name as it appears in the .prf file.

Net Names in the .lpf File For example, the following timing preferences appear in the .lpf file:

Figure 65:

```
MULTICYCLE FROM CLKNET "clk1_bn" TO CLKNET "clk2_dn" 6.000000
ns ;
INPUT_SETUP ALLPORTS 8.000000 ns CLKNET "clk1_cn" ;
INPUT_SETUP ALLPORTS 9.000000 ns CLKNET "clk2_dn" ;
CLOCK_TO_OUT ALLPORTS 10.000000 ns CLKNET "clk1_dn" ;
CLOCK_TO_OUT ALLPORTS 11.000000 ns CLKNET "clk2_bn" ;
```

Equivalent Net Names in the .prf File During the mapping process, nets get merged or optimized, and their equivalent nets are found and converted to the original preference. The preferences and equivalent nets then appear in the .prf file. For example:

Figure 66:

```
MULTICYCLE FROM CLKNET "clk1_in_c" TO CLKNET "clk2_in_c"
6.000000 ns ;
INPUT_SETUP ALLPORTS 8.000000 ns CLKNET "clk1_in_c" ;
INPUT_SETUP ALLPORTS 9.000000 ns CLKNET "clk2_in_c" ;
CLOCK_TO_OUT ALLPORTS 10.000000 ns CLKNET "clk1_in_c" ;
CLOCK_TO_OUT ALLPORTS 11.000000 ns CLKNET "clk2_in_c" ;
```

Net Names Matched in the Output Log To match up the net names shown in the .prf with those shown in the .lpf, open the **Output** log and scroll down to the “Optimizing” section. This section shows the net name that has been “clipped” for each preference and the new name that the preference has been moved to. For example:

Figure 67:

```
Optimizing...
Net clk1_bn is clipped. Preference MULTICYCLE on "clk1_bn" is moved to "clk1_in_c".
Net clk2_dn is clipped. Preference MULTICYCLE on "clk2_dn" is moved to "clk2_in_c".
Net clk2_dn is clipped. Preference INPUT_SETUP on "clk2_dn" is moved to "clk2_in_c".
Net clk2_dn is clipped. Preference USE SECONDARY on "clk2_dn" is moved to "clk2_in_c".
Net clk2_bn is clipped. Preference CLOCK_TO_OUT on "clk2_bn" is moved to "clk2_in_c".
Net clk1_dn is clipped. Preference CLOCK_TO_OUT on "clk1_dn" is moved to "clk1_in_c".
Net clk1_cn is clipped. Preference INPUT_SETUP on "clk1_cn" is moved to "clk1_in_c".
Net clk1_cn is clipped. Preference USE PRIMARY on "clk1_cn" is moved to "clk1_in_c".
```

Whenever the mapper does not find the equivalent net, it does not write a message to the Output file. Instead, it issues a warning in the Map Report file (.mrp). For example:

Figure 68:

```
WARNING: Net "net_a" is clipped. Preference INPUT_SETUP on
"net_a" is ignored.
```

The Physical Preference File

The physical preference file (.prf) is an ASCII output of the Map Design process. Preferences saved in the logical preference file (.lpf) are written to the native circuit description (.ncd) and to the .prf file during the map process. The map process rewrites the physical preference file each time it is executed. The .prf file is used as an input file to placement and routing and TRACE.

Note

A physical preference file should never be saved as an .lpf file, because a .prf file is quite different. A .prf file saved as an .lpf file will not perform as expected.

In the Diamond flow, the .prf file is considered a background file and is not visible in the File List. Though the .prf file can be opened from the project directory, its use for modifying preferences is no longer necessary and not recommended in most cases. You can use a Timing Preference File (.tpf), available from Timing Analysis View, to experiment with timing preferences and view the analysis without having to remap the design. But if you edit physical preferences in the .prf file, they will be inconsistent with logical ones in the .lpf, and they will be obsolete when mapping is rerun. It is better to define preferences in the logical preference file (.lpf) through the Diamond

preference views or with a text editor. This ensures that preference assignments in both the logical and physical domains stay in sync when the design is remapped.

Preferences from the .lpf file that have been modified or validated by map are listed inside the SCHEMATIC START and SHEMATIC END section, and they will be removed or replaced the next time the Map Design process is run. Preferences with syntax errors are preserved when the .prf file is read. Tools that write out preference files—for example, the Map Design process (map)—will automatically write the preserved preferences. Such a file, when read back in, will again produce syntax errors unless the erroneous preferences are corrected.

The .prf file can contain any number of preferences, both logical and physical, and any number of comments in any order. A comment consists of either a pound sign (#) or double slashes (//) followed by any number of other characters up to a new line. The program automatically comments out illegal preferences.

See Also ▶"Analyzing Static Timing" on page 694

- "Preferences" on page 1195
- "Setting Preferences" on page 459

Using Diamond's Preference Views

Diamond provides six views for editing design constraints known as preferences. These preference-editing views, which are available from the Diamond toolbar and Tools menu, include "Spreadsheet View" on page 408, "Package View" on page 426, "Device View" on page 431, "Netlist View" on page 432, "NCD View" on page 432, and "Floorplan View" on page 433.

Additionally, two read-only preference views are provided after placement and routing: "Physical View" on page 445, which enables you to view a more detailed layout of the design; and "Logic Block View" on page 447, which provides details of a selected block in schematic or tabular format.

Diamond's preference views enable you to develop constraints that will shorten turn-around time and achieve a design that conforms to critical circuit performance requirements. They enable you to perform the following tasks:

- Define timing constraints based on design signals and logic elements
- Define sysIO Buffer properties based on design signals
- Constrain clock signals to particular routing spines
- View, assign, and validate design signals and package pins
- Assign voltage reference pins
- Inspect the programming of design elements
- Group components and set REGIONs to help meet timing goals
- Cross-probe elements between physical and logical domains

- Perform DRC tests to ensure legal pin assignments or to detect incomplete or illegal conditions within the physical design database
- Generate simultaneous switching output (SSO) reports
- Back annotate post-route pin assignments to a logical preference file

See Also "Using the Zoom and Pan Tools" on page 454

Preference-Editing Views and Memory

Diamond provides six views for preference editing: Spreadsheet View, Device View, Package View, Netlist View, NCD View, and Floorplan View. Each of these views operates on preference access data that is stored in memory and shared across all six views. As soon as you create or modify a preference in one preference view, it is reflected immediately in all open preference views and in the status bar. When there are preference changes in memory, the tab of each open preference view is marked with an asterisk, and the status bar displays "Preferences Modified" on the bottom right.





Because of the shared memory between all preference-editing views, closing one or even all views does not delete the unsaved changes from memory, and you will not be prompted to save when you close a view. Diamond's status bar will always display "Preferences Modified" when you have unsaved preference changes, even when all preference views are closed.

Saving In-Memory Preference Changes You can save in-memory preference changes from any of the preference-editing views, with the exception of NCD View. When you issue the "Save" command, Diamond writes all of the preference changes stored in memory to the logical preference file, regardless of the preference views that were used for editing. The asterisks and the "Preferences Modified" indicators disappear as soon as you save the changes.

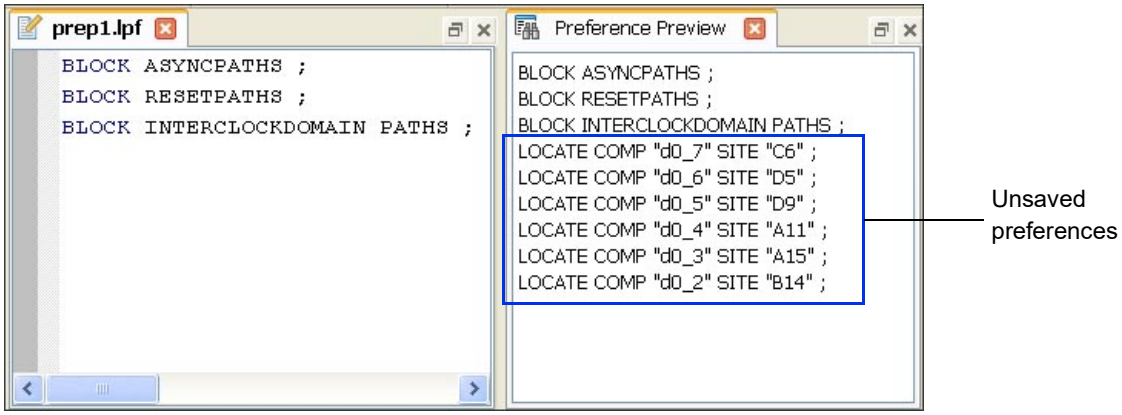
Diamond will prompt you to save in-memory preference changes each time you run a process, close the project, clear the tool memory, or exit Diamond.

Keeping Track of Unsaved Preferences Preference Preview is useful for keeping track of unsaved preference changes. Preference Preview lists all preferences that are already saved in the .lpf file, followed by the preferences that are in memory only. You can open the .lpf file, as well as Preference Preview, to do a side-by-side comparison.

See Also "Discarding In-Memory Preference Changes" on page 549

- "The Logical Preference File" on page 375
- "Previewing Preferences Before Saving" on page 547
- "Saving Preferences" on page 546

Figure 69:



Note

Preference Preview does not show unsaved changes that have been made manually to the .lpf file.

Spreadsheet View

Spreadsheet View provides a tabular format for viewing and assigning logical preferences. It displays all preferences, including those that have been assigned from other preference-editing views. It also displays timing constraints that have been set in the synthesis tool.

As soon as you have specified the target device, you can use Spreadsheet View to set global preferences. After synthesis and translation, you can use all of the following preference sheets to create and modify constraints:

Port Assignments The Port Assignments sheet provides a signal list of the design and shows any pin assignments that have been made. All ports assigned in the Logical Preference File (.lpf) and in the Native Generic Description (.ngd) file are displayed.

By default, the ports are grouped by direction—Input, Output, and Bidirectional—plus an "Others" category that includes user-defined port groups. This allows you to quickly focus, for example, on Output ports by closing the display of Input ports or vice versa. If you prefer to display the port rows in a single un-grouped alphabetical list, clear the "Show Group Row(s)" selection from the View > Display Group Row(s) menu.

The Port Assignments sheet enables you to assign or edit pin locations and other attributes by entering them directly on the spreadsheet. It also allows you to assign multiple signals, by right-clicking the selected signals and opening the Assign Pins dialog box. Any assignment problems are indicated

with a warning icon ⚠ in the signal Name column. The warning icon will appear for any invalid I/O assignment or combination.

Note

The default I/O type is LVCMOS25. Make sure this setting matches the IO in your design.

Pin Assignments The Pin Assignments sheet provides a pin list of the device and shows the signal assignments that have been made. In the Dual Function column, pins that can be used for I/O assignments or for another function, such as Vref, are identified.

By default, the rows of pins are grouped by bank number. You can also view the differential pin pair groupings within each bank by selecting "Show Differential Group Row(s)" from the View > Display Group Row(s) menu. To view the group of non-assignable pins, select "Show Non-user Assignable Pin Row(s)." If you prefer to display the rows of pins in a single un-grouped alphabetical list, clear the "Show Group Row(s)" selection from the View > Display Group Row(s) menu.

The Pin Assignments sheet enables you to edit signal assignments or assign new signals by right-clicking selected pins and opening the Assign Signals dialog box.

Clock Resource The Clock Resource sheet enables you to apply a clock domain to the device's primary or secondary clock or prohibit the use of primary and secondary clock resources to route the net. For LatticeECP2 devices, it enables you to use edge clock resources. For LatticeECP3 devices, it enables you to assign a secondary clock to a clock REGION that has been defined.

Route Priority The Route Priority sheet enables you to set the PRIORITIZE preference, which assigns a weighted importance to a net or bus. To set this preference, drag the desired nets from Netlist View to the Route Priority sheet. You can then select a priority value for each net. Values range from 0 to 100.

Cell Mapping The Cell Mapping sheet enables you to set the USE DIN and USE DOUT cell preferences for flip-flops in your design. The PIO Register column allows you to set the register to True or False. The True setting moves registers into the I/Os. The False setting moves registers out of the I/Os. To set these preferences, drag the desired registers from Netlist View to the Cell Mapping sheet.

Global Preferences The Global Preferences sheet enables you to set preferences that affect the entire design, such as junction temperature and voltage; BLOCK preferences applied to all paths of a particular type; and USERCODE. Also included in the Global sheet are sysCONFIG preferences for FPGA devices that support the sysCONFIG configuration port.

Timing Preferences The Timing Preferences sheet displays all timing preferences that have been set in the design, including BLOCK preferences

for specific nets, FREQUENCY, PERIOD, INPUT_SETUP, CLOCK_TO_OUT, MULTICYCLE, and MAXDELAY. It displays, in green font, the timing constraints that have been set in the synthesis tool and allows you to modify them and save the modified constraints to the .lpf file as preferences. You can create a new timing preference by double-clicking the preference name, which opens the dialog box. To modify an existing timing preference, double-click the preference name, edit the information in the dialog box, and click the Update button.

Group The Group sheet displays any groups that have been created and enables you to define a new cell, port, or ASIC group or create a new universal group (UGROUP). Double-click the group type to open the dialog box and create a new group preference. To modify an existing group preference, double-click the group name, edit the information in the group dialog box, and click the Update button.

Misc Preferences The Miscellaneous sheet enables you to define REGIONs, assign Vref locations, and reserve resources by setting a PROHIBIT preference. To set a new miscellaneous preference, double-click the preference type to open the dialog box. To modify an existing miscellaneous preference, double-click the preference name, edit the information in the dialog box, and click the Update button.

See Also
- "Assigning Signals" on page 466
- "Color Coding of Cell Values" on page 411
- "Setting Port, Net, and Cell Attributes" on page 475
- "Setting Timing Preferences" on page 492
- "Grouping Logical Components" on page 506
- "Working with Preference Sheets" on page 412
- "Migrating Pin Assignments" on page 484
- "Selecting a Device for Possible Pin Migration" on page 485
- "Viewing Incompatible Pins" on page 486

How Constraints are Recorded in Spreadsheet View

Spreadsheet View presents the union of constraints that originate from the HDL or schematic source or from the logical preference file (.lpf). Each cell of the preference sheet is color-coded to reflect the origin or status.

If any default, referenced, or HDL-based source constraint is modified, Diamond will add a preference to the .lpf file to record the change. Since preferences in the .lpf file have precedence over other constraint sources, the design mapper will ignore any conflicting constraints when the design is implemented. Diamond does not overwrite the original HDL or schematic constraint declaration. To restore the original constraint, you must delete the preference in the .lpf file.

Many preferences have one or more parameters. It is possible, therefore, to assign one parameter in the HDL or schematic source and another parameter

for the same preference in Spreadsheet View. Except for IOBUF attributes, Diamond will write the union of all parameters to the .lpf file. For IOBUF attributes that exist in the .lpf file and in the HDL for the same port or port group, those in the .lpf file will completely override those in the HDL. The exception to this rue is IO_TYPE and the placement constraint LOC, which will not get overriden.

For example, the IOBUF preference includes the parameter PULLMODE for an I/O port. Given a schematic-based IO_TYPE=LVCMOS12 attribute for the I/O port, if the PULLMODE parameter is modified within Spreadsheet View to PULLMODE=NONE, the following preference will appear in the .lpf file:

```
IOBUF PORT "my_signal" IO_TYPE=LVCMOS12 PULLMODE=NONE;
```

This IOBUF preference combines the IO_TYPE attribute from the source file with the new PULLMODE parameter that was made in Spreadsheet View.

See Also ▶"Conflicting HDL Attributes, Constraints, and Preferences" on page 401

▶ "Conflicting IOBUF Constraints" on page 398

Color Coding of Cell Values

The original font colors of cell values in Spreadsheet View are blue, gray, black, orange, green, and maroon. These colors, which can be changed in the Options dialog box, are based on the status of the cell values: default value, read-only value, modified (normal) value, referenced value, source value, or schematic value. The following descriptions explain the original color coding of these values.

- Blue font – Blue represents the default value of a preference. For example, the NONE value of PULLMODE for an HSTL I/O type is displayed in blue. Default values can be modified but not deleted.

  Blue also represents a value that is automatically assigned by the place & route process and written to the .ncd file. These values are displayed inside parentheses to distinguish them from default values.

- Gray font – Gray represents a read-only value, one that cannot immediately be changed. For example, the N/A value in the Vref column is displayed in gray when no Vref pin has yet been set or when Vref is not available for the selected I/O type. Gray also distinguishes pins that cannot be used as general-purpose I/Os.

- Black font – Black is the normal font color for a new or edited preference and represents a modified value, which can be a value that has been saved to the logical preference file (.lpf). For example, pin assignments made in the Port Assignments sheet and Block preferences made in the Global Preferences sheet are displayed in black. Modified values can be changed to the default value and font color, or to no value, by pressing the Delete key.

- Orange font – Orange represents a value for an element that is referenced by another preference. For example:

- If a port is a member of an IOBUF GROUP that has a Vref assignment, the port's Vref cell value will be displayed in orange unless it has been individually modified.
- If the Allports I/O Type is modified, the I/O type for all ports that have not been individually modified will be displayed in orange.

Referenced values can be modified but not deleted.

- Green font – Green represents the value from the design source. For Spreadsheet View, the source is the native generic database file (.ngd), which includes preferences from the HDL source code or EDIF file. The .ngd file also includes default configuration preferences, which appear in the Global Preferences sheet; and I/O types, which appear in the Port Assignments sheet. All of these source values from the .ngd file are displayed in green. Source values can be modified but not deleted.
- Maroon font – Maroon represents a value from a schematic source file. Schematic values can be modified but not deleted.
- Green font – Green represents a value from a timing constraint that was set in the synthesis tool. These SDC constraints are output to a synthesis .lpf file when the design is synthesized, and they are displayed in the Timing Preferences sheet when the strategy option "Use LPF Created from SDC in Project" has been enabled. By default, this option is set to True. Synthesis constraint values can be modified and saved as preferences, but not deleted. When you modify a synthesis constraint value and save it, the constraint is converted to a timing preference and saved to the active .lpf file.

Display of Incompatible Pins If you have selected a different device of the same family and package for possible pin migration, the rows of any incompatible pins will be dimmed on the Pin Assignments sheet.

See Also
- "Spreadsheet View" on page 408
- "Package View" on page 426
- "Netlist View" on page 432
- "Assigning Signals" on page 466
- "Migrating Pin Assignments" on page 484
- "Viewing Incompatible Pins" on page 486

Working with Preference Sheets

Spreadsheet View provides nine spreadsheets for setting and editing preferences. Preference values in spreadsheets are displayed in font colors based on their origin or status, and invalid values are highlighted in red.

The following techniques are useful for editing and viewing preferences in Spreadsheet View.

Enter data into table cells You can enter data into the table cells of Spreadsheet View in several ways:

- Double-click the cell that you want to edit.

  The cell changes to a drop-down menu or a text box, or it opens a dialog box, depending on the type of preference you are changing. You can then select from the menu, type a text entry, or select options and settings in the dialog box.

- Right-click the cell that you want to edit and choose from the pop-up menu. For a text entry, such as USERCODE, choose **Edit Cell** and type the string.

- To edit multiple cells at once, select the cells, right-click, and choose from the pop-up menu.

Cross-probe a signal assignment After a signal is assigned to a specific pin, you can cross-probe to Package View, Device View, Netlist View, or Floorplan View. After placement and routing, you can also cross-probe to NCD View.

To cross-probe a signal assignment:

- In the Port Assignments or Pin Assignments sheet, right-click the desired pin name or signal and choose the desired view from the **Show in** menu.

  The selected view is activated, and the pin is highlighted on the layout or design tree.

Sort a preference list You can sort a list on a preference sheet by a single column or more than one column.

By default, the ports on the Port Assignments sheet are sorted alphabetically within each group: input pins, output pins, bi-directional pins, and user-defined groups. Also by default, the pins on the Pin Assignments sheet are sorted alphabetically within each bank group. If you prefer to sort the rows based on a single un-grouped alphabetical list, clear the "Show Group Row(s)" selection from the View > Display Group Row(s) menu.

To sort a list by a single column:

- Select the desired preference sheet, and then click the heading of the column you want to sort.

  An up arrow is displayed in the heading, indicating that the list has been sorted by the column in ascending order. Click the heading again to sort the list in descending order.

To sort a list by more than one column:

1. In Spreadsheet View, select the desired preference sheet.
2. Choose **View >** **Sort**.
3. In the Sort dialog box, select the first column to sort by from the "Sort by" drop-down menu. Click **Ascending** or **Descending** to control the order of the sorted list.

4. Use the "Then by" boxes to select additional columns, as well as the **Ascending**/**Descending** option for each column. Click **More** to add more "Then by" menus if needed.

   The list will be sorted first by the column specified in the "Sort by" box and then by the additional columns in the "Then by" boxes in sequence.

5. Click **OK**.

Display an error or warning description Errors and warnings are highlighted with a triangular icon ⚠ inside a cell.

To display the error or warning description:

- Hold your mouse pointer over the cell that contains the icon.

  A description of the error is displayed in a tool tip.

Change the font of cell values You can change the font style and size of information that is displayed in the spreadsheet cells.

To change the font of spreadsheet cells:

1. Choose **Tools > Options**.
2. In the Options dialog box, select **Spreadsheet View**, and then select **Font**.
3. Click the **Sheet Font** button, which shows the current font and size.
4. In the Select Font dialog box, select the desired font, style, and size and click **OK**.
5. Click **OK** to close the Options dialog box.

Adjust the font color of cell values The font colors of values in preference cells are based on their status: default value, read-only value, modified (normal) value, referenced value, source value, or schematic value. Refer to "Color Coding of Cell Values" on page 411 for default colors and detailed explanations of these values.

To change the font color of cell values:

1. Choose **Tools > Options**.
2. In the Options dialog box, select **Spreadsheet View**, and then select **Color**.
3. Click the color box for the type of value you would like to change.
4. Choose the desired color from the Select Color dialog box and click **OK**.
5. Click **OK** to close the Options dialog box.

Adjust columns You can adjust column widths manually or choose the right-click command to fit the column to the text.

To adjust the column width manually:

- Drag the vertical border on the right of a column heading.

When you fit the column width to the text, all the text in the column is displayed.

To fit a column to the text, do one of the following:

- Double-click the vertical border on the right of a column heading.
- Right-click the desired column heading and choose **Fit Column**.

To fit all columns to the text:

- Choose **View > Column > Fit All Columns** or press the **F5** function key.

Rearrange the positions of columns You can move the position of any editable column on the Port Assignments Sheet or Pin Assignments sheet.

To rearrange the position of an editable column:

1. On the Port Assignments or Pin Assignments sheet, hold down the column's heading with your mouse pointer.
2. Drag the heading to the left or right to the desired position, and then release the mouse button.

Hide and re-display columns You can hide a single column or select several columns that you want displayed or hidden.

To hide a single column:

- Right-click the desired column heading and choose **Hide Column**.

To hide several columns:

1. Choose **View > Column > Visible Columns** or press **Shift+F5** to open the Visible Columns dialog box.
2. Do one of the following:
   - In the "Show these Columns list," select the columns that you want to hide, and then click the single-arrow button to move them to the "Hide these Columns" list.
   - Click the double-arrow button to move all columns to the "Hide these Columns" list, and then select those you want displayed by selecting them and moving them to the "Show these Columns" list.
3. Click **Ok**.

To re-display some of the hidden columns:

- Press **Shift+F5**, and move the hidden columns back to the "Shows these Columns list."

To re-display all columns:

- Choose **View > Column > Fit All Columns** or press the **F5** function key.

Auto complete When you type an entry into a cell in the I/O Types column, the cell presents a list of matching values. You can then:

- Choose the first (or only) matching value by pressing **Enter**.
- Choose an alternate matching value by using the up/down arrow keys and then pressing **Enter**.
- Continue typing the rest of the value. If no matching value is found, nothing will be entered.

Edit a sequence of cells in a column Spreadsheet View enables you edit the cells in a column sequentially without moving your mouse. It does this by automatically moving the focus to the next cell.

To edit a sequence of cells:

1. Enter the data in a cell and press **Enter**.

   The cell below the edited one is activated.

2. Proceed to edit the data in the active cell, press **Enter**, and continue to the next cell until you have finished editing the cells in the column.

   You can also pass to the next cell without editing the current one by pressing **Enter**.

Delete cell contents You can clear cells of their new or edited contents in the Port Assignments, Pin Assignments, Clock Resource and Global Preferences sheets. When you do this, Diamond empties the cells of their contents or redisplays the default or referenced values.

To delete the edited values from one or more cells:

- Select the cells that you want to clear and do one of the following:
  - Press the Delete key.
  - Right-click the selected cell and choose **Clear** from the pop-up menu.
  - Choose **Edit > Clear**.

  Diamond empties the cells of any values or re-displays the default or referenced values.

Delete row contents You can delete the assignments from one or more selected rows on the Port Assignments sheet. For each cell in the selected rows, Diamond empties the contents, or it re-displays the default, source, or referenced value.

To delete the assignments in one or more rows:

- Select the rows that you want to clear, right-click, and choose **Clear Selected Row(s)**. Alternatively, choose **Edit > Clear Selected Row(s)**.

Across the columns of the selected rows, Diamond empties some cells. For others, it reverts to a referenced, source, or default value, as follows:

- a referenced value defined in a port group or in All Ports
- a source value defined in the .ngd file
- a default value defined in the legal combination table

Cut, copy, and paste cell contents You can cut or copy cell contents and paste them into other cells in the same column of a Port Assignments, Cell Mapping, Clock Resource, or Route Priority sheet.

To copy or cut and paste cell contents:

1. Select the cells that you want to cut or copy.
2. Choose **Edit > Copy** or **Edit Cut** or use the **Ctrl+C** or **Ctrl+X** shortcut.
3. Select the first cell in the column where you want to begin pasting the cell contents.
4. Choose **Edit > Paste** or use the **Ctrl+V** shortcut.

Use fill down/fill up editing Fill Down/Fill Up editing is supported in Port Assignments, Cell Mapping, Clock Resource, and Route Priority sheets. In these preference sheets, you can fill a range of cells within the same column by dragging down across cells or by using the Fill Down command.

To fill a range of cells by dragging:

1. Highlight the source cell which contains a valid value, and then move your mouse to the lower-right corner of the cell. The cursor changes to a cross symbol.
2. Drag the cross down over the range you want to fill.

   The destination cells are filled with the same value as in the source cell.

   Or

   If you are working with the Pin Location column, you can drag down to increment the pin numbers in the destination cells.

To fill a range of cells by using the Fill Down command:

1. Highlight the source cell together with the range of destination cells below it. The source cell must be at the top of the selected range and contain a valid value.
2. Choose **Edit > Fill Down** or press **Ctrl+D**.

   The destination cells are filled with the same value as in the source cell.

   Or

   If you are working with the Pin Location column, the destination cells will be filled with incremental pin numbers following the number specified in the source cell.

See Also ▶"Changing the Grouped Display of Ports and Pins" on page 418

- "Filtering the Display of Ports and Pins" on page 419
- "Adding Custom Columns" on page 420

Changing the Grouped Display of Ports and Pins

By default, the rows of ports and pins in Spreadsheet View are grouped by category. This grouped display allows you to easily view one category by closing others. For example, you can focus on Bidirectional ports by closing the Input and Output categories. Simply click the arrow in front of a displayed category to open or close it. When you sort one or more columns, the ports and pins are sorted alphabetically within each group.

On the Port Assignments sheet, all ports that are defined in the native generic description file (.ngd) and the logical preference fie (.lpf) are displayed. The rows are grouped by Input, Output, Bidirectional, and by "Other," a category that includes user-defined PIO groups and "Nonexistent" ports. Nonexistent ports are ports that do not exist in the .ngd file, even though they are defined in the logical preference file. You can delete a row for a port that exists only in the.lpf file by right-clicking and choosing "Delete Selected Row(s)." You also have the option of viewing differential pair groups for all LVDS ports that have been defined in the source file.

On the Pin Assignments sheet, the rows are grouped by Bank number, and you have the option of viewing differential pin pair groups within each bank. You can also view all non-assignable pins as a single group. Because the pins are grouped by bank, the BANK column does not appear on the Pin Assignments sheet when the grouped display is turned on. But you can view the BANK column by turning off the grouped display.

To view differential pair groups:

- On the Port Assignments sheet or Pin Assignments sheet, choose **View > Display Group Row(s) > Show Differential Group Row(s)**.

  Any LVDS ports that have been defined in the source are now displayed in differential groups on the Port Assignments sheet, and the rows displayed on the Pin Assignments sheet are organized by differential pin-pair groups within each bank.

To view the group of non-assignable pins:

- On the Pin Assignments sheet, choose **View > Display Group Row(s) > Show Non User-Assignable Pin Rows**.

  The Non User-Assignable category is added to the bottom of the sheet and includes rows for all pins that are not user-assignable.

You can also turn off the grouped display of ports and pins at any time in order to view each list as a single un-grouped alphabetized list.

To turn off the grouped display of ports and pins:

- On the Port Assignments sheet or Pin Assignments sheet, choose **View > Display Group Row(s) > Show Group Row(s)**.

All ports and pins are now listed in a single un-grouped alphabetized list.

Each of the View > Display Group Row(s) commands acts as a toggle key to turn the grouped display on or off. When the display is turned on, the menu item shows a check mark in front of it.

See Also ▶"Filtering the Display of Ports and Pins" on page 419

▶ "Adding Custom Columns" on page 420

Filtering the Display of Ports and Pins

Both the Port Assignments sheet and the Pin Assignments sheet enable you to filter each column so that only the row contents that you specify are displayed.

To filter the display of ports and pins:

1. On the Port Assignments or Pin Assignments sheet, right-click any column heading and choose **Filter > Enable Filter**.

   Each column heading now appears with a downward-pointing arrow on the right.

2. Click the arrow for the column that you want to filter.

   The filter dialog box opens and displays a list of the items that appear in the row contents for that column. By default, all items are selected.

3. Do one of the following to filter the list:

   - Clear the individual items that you do not want to appear on the spreadsheet.
   - Clear the Select All selection at the top, and then select the types of items that you want to appear on the spreadsheet.

4. Click **OK**.

5. Repeat Steps 2-4 for other columns that you want to filter.

   When you have finished, the spreadsheet displays the filtered contents. The row numbers are displayed in blue font, and a filter button  appears in the heading of each filtered column.

To edit a column's filtered list:

- Click the filter button and modify the selections in the filter dialog box.

To disable the filtering for a single column:

- Click the filter button and choose  **Clear Filter From** <column name>.

To disable the filtering for all columns:

- Right-click any column heading and choose **Filter > Disable Filter**.

See Also ▶"Changing the Grouped Display of Ports and Pins" on page 418

▶ "Adding Custom Columns" on page 420

Adding Custom Columns

Spreadsheet View allows you to create custom columns on the Port Assignments sheet and the Pin Assignments sheet. Custom columns enable you to add your own information, such as notes for specific signals or pins or design data for third-party tools. You can create an unlimited number of custom columns, and you can include the column when you export to a Lattice CSV file or a Pin Layout File.

Custom columns can be added and edited at any stage of the design flow. Since custom columns are excluded from the generation of preferences, they do not cause a mapped implementation to get initialized back to pre-map status. The information is not added to the active logical preference file (.lpf) but is automatically saved with your project and made available for all implementations.

To add a custom column:

1. In Spreadsheet View, select the Port Assignments or Pin Assignments tab.
2. Right-click the column heading of any editable column and choose **Insert Custom Column**.
3. In the Custom Column dialog box, type a name for the column. The name must be unique to the selected preference sheet and must not conflict with a column heading that is built in to the Port Assignments sheet or the Pin Assignments sheets.

   Note

   Because the Port Assignments and Pin Assignments sheets act independently of one another, it is possible to use the same name for a custom column on each of these sheets. When you later export a pin layout file, Diamond will append "_port" or "_pin" to distinguish each custom column.

4. Click **OK**.

   The custom column appears on the spreadsheet. The column heading name is displayed in italic font to distinguish it as a custom column.
5. Enter the desired information into the cells of the custom column as desired.

   The custom column information is automatically saved with the project and is available for each implementation.

Editing a Custom Column You can edit or delete cell contents of a custom column, rename the heading, or remove the entire column.

To edit or delete a cell's contents:

▶ Double-click a cell, and then type a new value or press the Delete key.

To edit a custom column heading:

1. Right-click the custom column heading and choose **Edit Custom Column**.
2. Type a new name in the dialog box and click **OK**.

To remove a custom column:

- Right-click the custom column heading and choose **Remove Custom Column**.

See Also "Working with Preference Sheets" on page 412

- "Exporting a Pin Layout File" on page 564

Searching Preference Sheets

Spreadsheet View provides Find dialog boxes for the Port Assignments, Pin Assignments, Clock Resource, Route Priority, and Cell Mapping preference sheets. Except for Pin Assignments, each of these preference sheets enables you to replace the contents of located preference cells.

Searching Port or Pin Assignment Sheets The Port Assignments and Pin Assignments sheets enable you to locate any string, including a preference, a pin assignment, and any built-in value such as a pad name. The dialog box for the Port Assignments sheet includes options for both finding and replacing.

To search for a string on the Port or Pin Assignments sheet:

1. Select the Port Assignments or Pin Assignments tab, and then do one of the following:
   - Click a cell that contains the cell value you want to find, and then choose **Edit > Find** or press **Ctrl+F**.

     The Find dialog box is populated with the cell value.
   - Click an empty cell and choose **Edit > Find** or press **Ctrl+F** to open the Find dialog box, and then type the name of the signal, pin, keyword, or other string you want to locate. Type the first few characters of the name to locate multiple items.
2. Select the "Match case" or "Match whole word" options as desired.
3. Click **Find Next**.

   Afterwards, you can continue to click the Find Next button or press the Enter key to locate each instance.
4. To search back toward the top, select Search Up and click **Find Next** or press **Enter**.
5. If you are using the Port Assignments sheet and the items you are locating are user preferences or keywords that you want to replace, do the following:

a. Click the **Replace** button at the top of the dialog box, and then enter the desired replacement preference or keyword in the "Replace with" text box.

b. Click the **Replace** button at the bottom of the dialog box to replace the next found item. To replace all found instances, click **Replace All**.

Searching Clock Resource, Route Priority, and Cell Mapping Sheets
Use the Find dialog boxes to locate a clock preference keyword, a priority number for prioritized nets, or keywords for registers used as flip-flops.

To find a cell value without replacing it:

- Choose **Edit > Find** or press **Ctrl+F**.

To find and replace a cell value:

1. Do one of the following:
   - Select the cell that contains the value you want to find and choose **Edit > Replace** or press **Ctrl+H**.

     The value appears in the "Find what" text box of the dialog box.
   - Choose **Edit > Replace** or press **Ctrl+H** and type the cell value that you want to find.
2. In the "Replace with" text box, type the value that you want to use as replacement, and then do one of the following:
   - To replace all cells that contain the "Find what" value, click **Replace All**.
   - To replace some but not all cells that contain the "Find what" value, click **Find Next** until the cell is highlighted that you want replaced, and then click **Replace**. Repeat for each cell that you want replaced.

Displaying I/O Placement Assignments in Spreadsheet View

After placement and routing, you have the option of viewing or hiding all the assignments that have been written to the native circuit description (.ncd) file. This includes automatic assignments made by Place & Route (PAR) and assignments that originated from the HDL source or the logical preference file (.lpf). By default, the display of I/O Placement assignments from the .ncd is turned on in Spreadsheet View.

To display or hide .ncd assignments:

- Choose **View > Display IO Placement** or click the Display IO Placement button  on the toolbar.

  This command acts as a toggle key and turns the display of I/O placement assignments on or off.

When "Display IO Placement" is turned on, NCD assignments are displayed differently, depending on whether they originated from a source or constraint file or were assigned by the placement and routing process.

Placement Assignments Originating from a Source or Constraint File Placement assignments from the .lpf file appear in black, or in the color designated for "Normal Value" in the Options dialog box. Those from the HDL source appear in green or in the color designated for "Source Value" in the Options dialog box. But unlike the display of these values before placement and routing, each assignment is now displayed twice: once normally and once in parentheses. The value in parentheses represents the value written to the native circuit description file (.ncd). When you change a post-PAR value, the new value, followed by the .ncd value in parentheses, is displayed. When you save the change, all values in parentheses are cleared.

Placement Assignments Made by PAR Placement assignments made by the Place & Route process (PAR) are displayed in blue font or in the color designated for "Default Value" in the Options dialog box. Pin and bank assignments are displayed in parentheses and appear only once. All other assignments are displayed twice: once normally and once in parentheses. When you edit the value for one of these assignments, the font color is changed to black, or the "Normal Value" color. The value you assigned then appears without parentheses, followed by the PAR-assigned value in parentheses.

When you edit preferences after placement and routing and save them, the PAR-generated assignments are cleared from Spreadsheet View and only the assignments from the source or .lpf file are displayed. The "Display IO Placement" command is then unavailable until you rerun Place & Route.

Hiding Default Values

You have the option of hiding the default values shown on the Port Assignments, Pin Assignments, and Global Preferences sheets. Hiding the default values narrows the display to preference, source, and read-only values. When turned on, default values are displayed in blue font, or in the font color that you have designated for "Default Values" in the Options dialog box. You can hide them and display them again at any time.

To hide or display default values:

- Choose **View > Show Default Value**

  This command acts as a toggle key and turns the display of default values on or off. A check mark indicates that they are to be displayed.

After placement and routing, if "Display IO Placement" is enabled, all assignments written to the .ncd file by Place & Route (PAR) are displayed in the default font color on the Port Assignments and Pin Assignments sheets. Each PAR-assigned value is displayed inside parentheses to distinguish it from a default value.

Adding Preferences from the Toolbar

You can quickly add timing preferences, a grouping assignment, or a Vref location by clicking the appropriate preference button on the toolbar. After you add the new preference, it is displayed in the appropriate preference sheet.

To add a new preference using the Spreadsheet View Toolbar:

1. Click the preference toolbar button for the preference you want to add:

    adds a BLOCK preference.

    adds a PERIOD or FREQUENCY preference.

    adds an INPUT_SETUP or CLOCK_TO_OUT preference.

    adds a MULTICYCLE preference.

    adds a MAXDELAY preference.

    adds a DEFINE GROUP preference for a cell, port, or ASIC group.

    adds a LOCATE VREF preference.

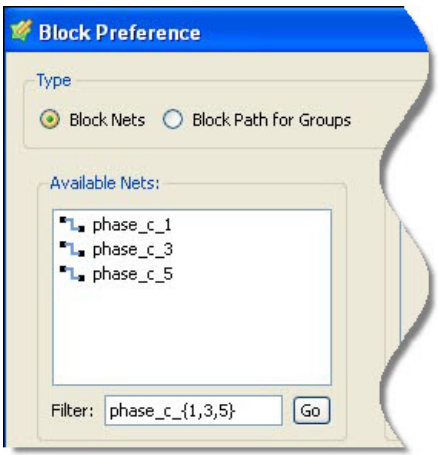
2. In the preference dialog box, specify the parameters and click **Add** or **OK**.

   The new preference is displayed in the Timing, Group, or Misc preference sheet.

Using Filters and Wildcards

The preference dialog boxes of Spreadsheet View provide a quick way to locate and filter design elements such as PORT, NET, ASIC PIN, and CELL objects. Many of the filters in these dialog boxes enable you to use wildcards to specify multiple elements that you want included in a single preference. Wildcards can be especially useful in large designs, because they reduce the number of preferences and conserve memory.

In the following example, the expression `phase_c_{1,3,5}` is typed in the Block Preference filter. When the Go button is clicked, only the three nets in the Available Nets list is displayed.



Clicking OK adds the preference with the wildcard expression as a single item to the existing BLOCK Preference list. It will be added as a single preference to the logical preference file with the Save command.

Note

Wildcard notation is not available with all preference parameters. See the individual preference descriptions in the "Constraints Reference Guide" on page 1194 for wildcard support.

Filter and Wildcard Support in Spreadsheet View Filters and wildcard expressions are supported in the following preference dialog boxes:

- Block
- Group Assignment
- INPUT_SETUP/CLOCK_TO_OUT
- MAXDELAY
- MULTICYCLE
- PERIOD/FREQUENCY

Wildcard Expressions The following basic wildcard expressions can be used with filters:

***** Match any (zero or more) characters.

? Match any single character.

?+ Match one or more characters.

{...} Match the enclosed list or range. The range can include a string of characters separated by commas, a lexical range separated by a dash (-), or an integer range separated by a colon (:).

{...}+ Match one or more occurrences of any character from the preceding list.

For more detailed information about wildcard expressions and syntax, see "Using Wildcard Expressions in Preferences" on page 533.

Package View

Package View shows the pin layout of the device and displays the assignments of signals to device pins. Package View interacts with Netlist View for assigning pins, enabling you to drag selected signals to the desired locations on the pin layout to establish LOCATE preferences. Each pin that is assigned with a LOCATE preference is color-coded to indicate the port direction of the related signal port. Package View allows you to edit these assignments, and it allows you to reserve sites on the layout that you want to exclude from placement and routing.

Banks and pin types are differentiated by color on the layout. You can use the Color Legend dialog box, available from the Package View toolbar, to view and change the default color coding. You can also use the Pin Display Selection dialog box to filter the display so that only certain banks and pin types are shown.

As you move your mouse pointer slowly over the layout, pin descriptions and locations are displayed in tool tips and in the status bar. The "View > Show Differential Pairs" command displays fly wires between differential pin pairs and identifies the positive differential pins with a magenta-colored + indicator.

Package View is available as soon as the target device has been specified.

See Also
- "Assigning Signals Using Package and Netlist Views" on page 471
- "Modifying Signal Assignments in Package View" on page 472
- "Migrating Pin Assignments" on page 484
- "Selecting a Device for Possible Pin Migration" on page 485
- "Displaying Incompatible Pins" on page 428
- "Using the Zoom and Pan Tools" on page 454

Using the "Show" Commands

The View menu in Package View provides two groups of "Show" commands. The first group consists of commands that are mutually exclusive; when one command is activated, the others become inactive. The second group consists of commands that are not mutually exclusive; they can be activated at the same time that another "Show" command is active.

In addition to the View menu, the "Show" commands are available from the pop-up menu when you right-click a blank space on the Package View layout.

Mutually Exclusive Commands When one of the following commands is activated, the other three are made inactive:

- Show Bank IO

- Show Port Groups
- Show DQS Groups
- Show SSO View

Non-mutually Exclusive Commands The following commands can be activated while another "Show" command is active:

- Show Pin Display Selection Dialog
- Show Differential Pairs
- Show Incompatible Pins
- Show Placement

Displaying Bank I/O Assignments

By default, Package View displays any assigned I/Os within each color-coded bank. The bank I/O display is turned off, though, when the display for Port groups or DQS groups is turned on.

To display bank I/Os:

- In Package View, choose **View > Show Bank IO**

  The I/O assignments are displayed within each bank. Port groups, DQS groups, and SSO View are hidden.

See Also "Displaying Incompatible Pins" on page 428

Displaying Port Groups

Package View enables you to view the placement of assigned port groups on the pin layout. Each port group is color-highlighted with the same distinctive color that is displayed in the Group sheet of Spreadsheet View.

To display port groups:

- In Package View, choose **View > Show Port Groups**.

  The color-coded port groups are displayed. Bank I/O assignments, DQS groups, and SSO View are hidden.

Displaying DQS Groups

You can view the placement of DQS groups in color-coded format on Package View's pin layout by turning on the "Show DQS Groups" command.

To display DQS Groups:

- In Package View, choose **View > Show DQS Groups**.

The color background of each DQS group is displayed. Bank I/O assignments, port groups, and SSO View are hidden.

Displaying SSO View

The "Show SSO View" command, available from Package View, displays the output pins that have been selected for SSO analysis. Tool tips for each pin provide SSO details. Output pins that passed SSO analysis are color-coded green; those that failed are color-coded red. This command also opens the SSO report in the Output window.

To display the SSO View:

- In Package View, choose **View > Show SSO View**.

  The SSO analysis output pins are displayed, color-coded according to pass or fail status. Bank I/O assignments, port groups, and DQS groups are hidden. The tool tip for each SSO pin includes information from the SSO Analysis Report.

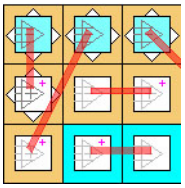
Displaying Differential Pin Pairs

Package View's "Show Differential Pairs" command enables you to identify all the differential pin pairs on the layout. When this command is activated, fly wires are displayed that connect the differential pin pairs. Each unplaced positive differential pin displays a small magenta-colored cross in the upper right corner. By default, these indicators are hidden.

To display differential pin pairs:

- In Package View, choose **View > Show Differential Pairs**.

  The layout displays the fly wires connecting differential pins and the cross indicator for each positive differential pin.



  The "Show Differential Pairs" command acts as a toggle key and displays a check mark when the differential pair fly wires are displayed. To turn off the display, select the command again to remove the check mark.

Displaying Incompatible Pins

The "Show Incompatible Pins" command opens a dialog box that enables you to select one or more devices of the same family and package and then view the pins on the layout that would not be available or would have a different function were you to switch devices.

To display incompatible pins on the pin layout:

- In Package View, choose **View > Show Incompatible Pin**, and in the dialog box, select one or more devices and click **OK**.

  Package View displays each incompatible pin with a dark circle behind the pin square.



See Also "Migrating Pin Assignments" on page 484

"Selecting a Device for Possible Pin Migration" on page 485

Displaying I/O Placement Assignments in Package View

After placement and routing, Package View enables you to examine all pin assignments that have been written to the native circuit description file (.ncd), including those made by Place & Route (PAR). A rotated square in the background of each placed pin serves as a placement indicator when the "Display IO Placement" command is turned on. The pin assignments that originated from the logical preference file (.lpf) are displayed with a color fill to distinguish them from PAR-assigned pins.

To display I/O Placement assignments in Package View:

- In Package View, choose **View > Display IO Placement** or click the Display IO Placement button  on the toolbar.

  A rotated square appears in the background of each placed pin. PAR-assigned pins are displayed with no color fill.



  Pin assignments that originated from the .lpf file are displayed with a color fill.



The "Display IO Placement" command acts as a toggle key and displays a check mark when the assignments are displayed. To turn off the display, select the command again to remove the check mark.

When you edit pin assignments after placement and routing and save them, the placement indicators are cleared from Package View. The "Display IO Placement" command is then unavailable until you rerun Place & Route.

Changing the Default Color Coding

The color coding in Package View is used to differentiate each bank and each type of pin. You can easily view the significance of each color by opening the Color Legend, which is available on the Package View toolbar. The Color Legend dialog box also enables you to change the default color coding for banks and pin types.

To change the default color coding for banks and pins:

1. Click the Color Legend button  on the Package View toolbar.
2. In the dialog box, select the Banks or Pins tab.
3. Click the color box in front of the desired bank or pin type that you want to change.
4. In the Select Color dialog box, select a different color and click **OK**.
5. If you want to change more than one color, click **Apply** in the Color Legend dialog box, and then repeat Steps 3 and 4. To revert to the default colors, click **Restore Defaults**.
6. When you have finished color editing, click **Close**.

Filtering the Display of Banks and Pin Types

When you open Package View, it shows all banks and pin types by default. You can use the Pin Display Selection dialog box to filter the banks and pin types so that only selected banks and pin types are displayed.

To filter the display of banks and pin types in Package View:

1. In Package View, choose **View > Show Pin Display Selection Dialog** or click the Pin Display Selection button  on the toolbar.
2. Do one of the following:
   - To eliminate the display of a few items, clear the check mark in front of each item that you do not want displayed.
   - To filter the display so that just a few items are displayed, click **Clear All**, and then select the banks and pin types that you want to see displayed. At least one bank must be selected.

     For example, to display only the dual function pins that are available in Bank 5, you would click **Clear All**, and then you would select **Bank 5** in the Visible Banks section and **Dual Function** in the General Purpose section.
3. Click **Close**.

   Package View displays only the items that you selected.

To re-display all banks and pin types:

- Reopen the Pin Display Selection dialog box, click **Select All**, and then click **Close**.

Package View displays all items.

When you close Package View and re-open it, it automatically reverts to the default view and displays all banks and pin types.

See Also ▶"Using the "Show" Commands" on page 426

Reversing the Pin Layout View

You can reverse the view of Package View 's device pin layout from a bottom to a top perspective or vice versa.

To reverse the pin layout view:

- In Package View, choose **View > Top View** or **View > Bottom View**.

  The View command acts as a toggle key and changes to its opposite, depending on the current view.

Device View

Device View provides an alphabetized index of device resources based on the target device. Statistics in Device View include:

- DCS blocks
- Assignable PIO cells, categorized by bank
- IOLOGIC sites
- PFF and PFU slices
- DQS blocks
- PCS blocks
- sysDSP blocks
- sysMEM blocks
- PLL and DLL blocks
- DDR support with bonded DQS, categorized by bank
- Other embedded ASIC blocks.

The right-click menu of Device View enables you to assign PROHIBIT preferences for sites that you want to exclude from placement and routing. From Device View, you can cross-probe selected resources to their sites in Package View, Floorplan View, and Physical View.

Device View is available as soon as the target device has been specified.

See Also ▶"Reserving Sites" on page 524

Netlist View

Netlist View displays the design elements of the post-synthesis native generic database (NGD) netlist. The NGD is a binary speed-optimized data structure that is used by the system to browse the logical netlist. Netlist View organizes the netlist by ports, instances, and nets, and it provides a toolbar button and design tree view for each of these categories. Each design tree view is equipped with utilities for filtering the list and searching for elements. Netlist View enables you to set location, timing, and group preferences.



Ports design tree

Right-click selected signals to assign pin locations, create port groups, or assign INPUT_SETUP and CLOCK_O_OUT preferences.

Drag selected signals to Package View to assign them on the pin layout.



Instances design tree

Right-click selected instances to create UGROUPs. Right-click selected registers to create cell groups or UGROUPs.

Drag registers to the Cell Mapping sheet of Spreadsheet View to specify registers for flip-flops.



Nets design tree

Right-click a selected clock net to set a PERIOD or FREQUENCY preference. Right-click a selected net to set a BLOCK preference.

Drag selected nets to Spreadsheet View's Route Priority sheet to prioritize them.

Netlist View is available after synthesis and translation.

See Also
- "Modifying Pin Assignments Using Netlist View" on page 470
- "Assigning Signals Using Package and Netlist Views" on page 471
- "Creating UGROUPs" on page 509
- "Setting Timing Preferences in Netlist View" on page 503
- "Setting LOCATE Preferences in Floorplan View" on page 488

NCD View

NCD View, available after a successful run of Place & Route, provides a categorized index of synthesized design resources and consumption based on the target device and the in-memory native circuit description (NCD) database. Statistics cover System PIOs, User PIOs, PFUs, PFFs, sysDSP blocks, sysMEM blocks, IOLOGIC, PLL/DLLs, and other embedded ASIC blocks.

NCD View is organized by nets and instances and provides a toolbar button and design tree view for each of these categories. Each design tree view is equipped with utilities for filtering the list and searching for elements.

Instances design tree

Right-click selected instances to create a new UGROUP or to view logic details in Logic Block View.

Cross-probe selected instances to Netlist View, Floorplan View, or Physical View; and cross-probe PIO instances to Package View.

Nets design tree

Right-click a selected net to view the connection in Floorplan View or the complete physical path in Physical View

See Also "Viewing Logic Details of Components" on page 448

"Creating UGROUPs" on page 509

Floorplan View

Floorplan View provides a large-component layout of your design. It displays user constraints from the logical preference file (.lpf) and placement and routing information. All connections are displayed as fly-lines. Floorplan View allows you to create REGIONs and bounding boxes for UGROUPs and specify the types of components and connections to be displayed. As you move your mouse pointer slowly over the floorplan layout, details are displayed in tool tips: the number of resources for each UGROUP and REGION; the number of utilized slices for each PLC component; and the name and location of each component, port, net, and site.

If your design contains partitions, for the incremental design flow, you can display and edit the partitions in Floorplan View. See "Floorplan View and Physical View Partition Support" on page 674 for details.

Floorplan View is available as soon as the target device has been specified.

See Also "Creating UGROUPs" on page 509

"Creating a REGION" on page 519

"Setting LOCATE Preferences in Floorplan View" on page 488

"Viewing Logic Details of Components" on page 448

"Using Incremental Design Flow" on page 651

"Using the Zoom and Pan Tools" on page 454

Displaying Assignments and Connections

The Floorplan View toolbar allows you to select the types of elements that will be displayed on the layout, including REGIONs, UGROUPs, placed components, and component connections. These commands are also available from the View menu.

Display Placement – displays all placed components, including assignments originating from the logical preference file (.lpf) and those made

by Place & Route. The Display Placement button controls the Display Placement Group button, the Display Congestion button, and the three Display Connection buttons below it. It must be active in order for group member components, routing congestion, and component connections to be displayed.

 Display Placement Groups – highlights the placed UGROUP component sites with the designated color fill.

 Display Congestion – Displays a representation of the amount of routing congestion for a PLC component or site. When Display Congestion is turned on, a highlighted rectangle overlays the site. The thickness of the rectangle represents the percentage of congestion for the row and column location. The tool tip for the location shows the precise percentage of congestion.

 Display Connections To/From Selection(s) – This command controls the two Display Connections buttons below it. When active, it enables you to activate one or both of these buttons to view the connections between selected components, outside of selected components, or both:

 Display Connections Between Selection(s) – shows the connections between selected components.

 Display Connections Outside of Selection(s) – shows the connections that extend outside of the selected components.

 Display Placement Preferences – shows placement assignments that originated from the .lpf file. When this button is active, placed components assigned with a LOCATE preference are distinguished with a cross-hatch pattern.

Note

The Display Placement Preferences button must be active in order to access the editing commands for creating or editing a UGROUP or REGION, prohibiting a site, or setting a LOCATE preference.

The Display Placement Preferences button also controls the three buttons below it. These buttons enable you to view REGIONs and UGROUPs and the logical connections between them.

 Display REGIONs – shows the bounding boxes of REGIONs that exist in memory or in the logical preference file.

 Display UGROUPs – shows the bounding boxes of UGROUPs that exist in memory or in the logical preference file.

 Display Logical Connections – shows the logical connections between UGROUPs, including those assigned to REGIONs, and between

UGROUPs and assigned PIO sites. The connections are based on the logical instances within each UGROUP.

Note

Connections in Floorplan View are displayed at a high level of abstraction and do not reflect the complete physical path. When you cross-probe a delay path from Timing Analysis View, Floorplan View displays the delay path connection, but it does not enable you to cross-probe the path to other views. To examine the physical path, cross-probe from Timing Analysis View to Physical View instead of Floorplan View.

Displays partitions that are included in an incremental design flow.

See Also

- "Using the Zoom and Pan Tools" on page 454
- "How Assignments are Displayed in Floorplan View" on page 435
- "Display of FPGA Elements" on page 440
- "Using Incremental Design Flow" on page 651

How Assignments are Displayed in Floorplan View

Floorplan View displays assignments from the logical preference file (.lpf) at any stage of the design flow. These can include assigned sites, UGROUPs, REGIONs, and reserved resources. After routing, it displays all placements assigned by the Place & Route process, as well as assignments originating from the .lpf file.

The "Display" commands on the toolbar or the View menu control the display of assignments. The types of assignments displayed will depend on whether the commands "Display Placement," "Display Placement Preferences," or both are activated.

Assigned Sites After placement and routing, a site to which a component has been assigned appears on the layout with a solid fill.

Figure 70:



If the assignment originated from the logical preference file or the HDL source, the site will be filled and overlaid with a cross-hatch pattern when both "Display Placement" and "Display Placement Preferences" are activated.

Figure 71:



When only "Display Placement Preferences" is activated, an assignment originating from the logical preference file or HDL source is displayed with the cross-hatch pattern but without the fill.

Figure 72:



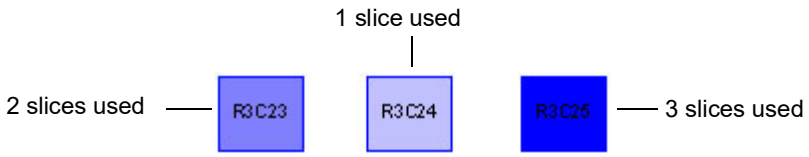
Reserved Sites Sites reserved with the PROHIBIT preference are displayed with a gray pattern.

Figure 73:



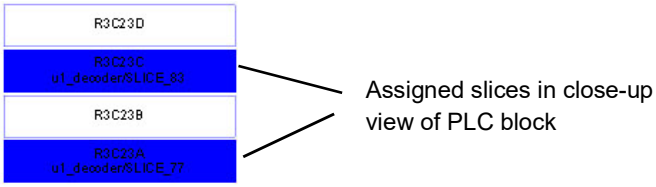
PLC Blocks (PFU and PFF) After placement and routing, a programmable logic cell (PLC) in which one or more logical components have been placed will appear on the layout with a solid fill until you zoom in. The color fill will be darker or lighter in shade, depending on how many sites are used.

Figure 74:



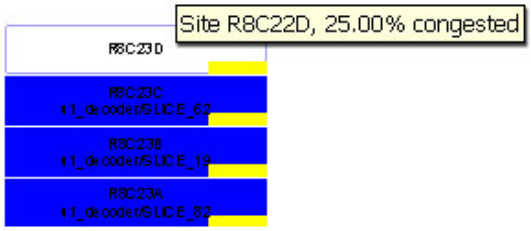
As you zoom in so that individual slices are visible, only the slices that contain placed logical components are displayed with a color fill.

Figure 75:



When Display Congestion is turned on, a highlighted rectangle is displayed that represents the amount of congestion for the row and column location. The tool tip shows the precise percentage of congestion.

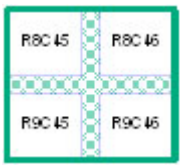
Figure 76:



UGROUPs Anchored UGROUPs are displayed when both the "Display Placement Preferences" and "Display UGROUPs" commands are activated. An anchored unbounded UGROUP is indicated with a small square icon at the upper left of the assigned anchor location. The icon becomes visible as you zoom in. 

A bounded UGROUP is displayed with a solid border, corresponding to the assigned width and height, and a dotted pattern between blocks. UGROUPs are distinguished from one another by their designated colors. After placement, the sites that contain the member elements of the UGROUP are filled with the designated color when the "Display Placement Groups" command is activated.

Figure 77:



Bounded UGROUP



Placed UGROUP members

Note

To quickly view the color legend for all UGROUPs or REGIONS in the design, click the color legend button  on the toolbar, and then select the UGROUPs or REGIONs tab.

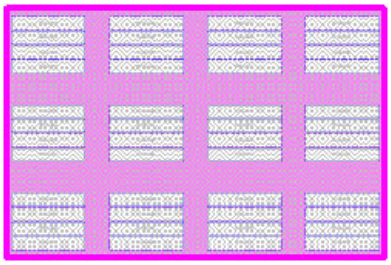
REGIONs REGIONs are displayed when both the "Display Placement Preferences" and "Display REGIONs" commands are activated. A REGION bounding box is shown with a solid border and a solid fill between blocks. It is distinguished from other REGIONs by its designated color. When a REGION is reserved with the Prohibit command, it is overlaid with a gray dotted pattern.

See Also ▶"Displaying Assignments and Connections" on page 433

Figure 78:



REGION



Reserved REGION

- "Assigning Signals" on page 466
- "Creating UGROUPs" on page 509
- "Creating a REGION" on page 519
- "Reserving Sites" on page 524
- "Reserving REGIONs" on page 525
- "Setting LOCATE Preferences in Floorplan View" on page 488
- "Floorplan View and Physical View Partition Support" on page 674

Viewing and Changing the Color Coding

The Color Legend dialog box in Floorplan View provides a color map of items such as sites and connections, plus all UGROUPs and REGIONs that are in the design. It enables you to quickly edit the color of each of these items. You can also change the color of a UGROUP or REGION by using the pop-up menu on the floorplan layout.

To view and edit the color coding:

1. Click the Color Legend button on the toolbar.

   The Floorplan View Color Legend shows three tabs: Item Types, which displays the colors of connections, selected items, and types of sites; UGROUPs, which shows the color coding of each UGROUP preference in the design; and REGIONs, which shows the color coding of each REGION preference in the design.
2. Select the tab for the category that you want to view or change.
3. Click the color box in front of the UGROUP, REGION, or item type that you want to change.
4. Select a color from the Select Color dialog box and click **OK**.
5. If you want to change more than one color, click **Apply** in the Color Legend dialog box, and then repeat Steps 3and 4. To revert to the default colors for sites or connections, click **Restore Defaults**.
6. When you have finished color editing, click **Close**.

To change the color of a UGROUP or REGION from the pop-up menu:

1. On the floorplan layout, select the UGROUP or REGION that you want to change.
2. Right-click the selection and choose **Edit Color**. Select a color from the Select Color dialog box and click **OK**.

Going to a Specific Site

Floorplan View provides a "Go To Location" dialog box for quick navigation to a specific column and row on the layout.

To go to a specific site:

1. In Floorplan View, right-click an empty space on the layout and choose **Go to Location**.
2. In the dialog box, enter the desired Row and Column.
3. Click **OK**.

   Floorplan View zooms in to the location you specified.

See Also ▶"Using the Zoom and Pan Tools" on page 454

▶ "Searching for Design Elements" on page 455

Display of FPGA Elements

The following table describes the graphical representations of FPGA component blocks and ports that are visible on the floorplan layout. These elements can be cross-probed between Device View's categorized list and Floorplan View. For complete information about FPGA logic design primitives, see the "FPGA Libraries Reference Guide" on page 1665.

Note

FPGA elements that are specific to certain devices are so noted below the description.

Table 20: FPGA Elements

Object	Assignable	Description	Location syntax
ALU	Yes	Arithmetic Logic Unit, configurable as a 24-bit or 54-bit ternary adder/subtractor. *ECP5, LatticeECP3, LIFMD/F*	R<row> C<column>
AMBOOT	Yes	Automatic multiple boot that enables the LatticeECP3 device to reconfigure itself upon SED failure. *LatticeECP3*	AMBOOT
CIBTEST	No	Common interface block test. CIBTEST blocks are used for internal hardware testing only. CIBTESTs are positioned around the chip and in rows near special blocks such as EBRs, MULTs, and MACs.	
CLKDIV	Yes	Clock divider block. *ECP5, LatticeECP2/3, LatticeSC/M, LatticeXP2, LIFMD/F, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	LatticeECP2/3 R<row> C<column> LatticeSC/M CLKDIV1A, CLKDIV1B, CLKDIV1C, CLKDIV1D MachXO2 TCLKDIV0, TCLKDIV1 BCLKDIV0, BCLKDIV1
CLKDIVTEST	No	Clock divider block test. Used for internal hardware testing only.	
CLKFBBUF	Yes	Clock feedback buffer that provides a dummy feedback delay between PLL clk output and PLL fb port. Two are provided per device (1k and above) *MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	CLKFBBUF0 CLKFBBUF1
DCC	No	Dynamic Quadrant Clock Enable/Disable. For internal use only. *ECP5, LatticeECP3, LIFMD/F, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	

Table 20: FPGA Elements

Object	Assignable	Description	Location syntax
DCM	Yes	Dynamic clock MUX incorporating a multiplexer function, for primary clock tree bank 6 and 7. *MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	DCM6, DCM7
DCS	Yes	Dynamic clock select, a global clock buffer incorporating a smart multiplexer function. The DCS takes two independent input clock sources and avoids glitches or runt pulses on the output clock regardless of when the enable signal is turned on. *ECP5, LatticeEC, LatticeECP/2/M, LatticeECP3, LatticeSC/M, LatticeXP/2, LIFMD/F*	For most devices: ULDC0, ULDC1 URDCS0, URDCS1 LLDCS0, LLDCS1 LRDCS0, LRDCS1 Example: "M_DCS" SITE "LLDCS1" where: UL is upper left UR is upper right LL is lower left LR is lower right For LatticeSC and LatticeSCM devices: DCSTA, DCSTB DCSLA, DCSLB DCSRA, DCSRB DCSBA, DCSBB Example: "U_DCS" SITE "DCSTA" where: DCST is top DCSL is left DCSR is right DCSB is bottom
DLL	Yes	Delay-locked loop, a digital circuit used to perform clock management functions on and off the chip.	R<row> C<column> For LatticeSC and LatticeSCM devices: DLL_[UL][UR][LL][LR]C[CDEF] Example: DLL_LLCC where: UL is upper left, UR is upper right, LL is lower left, LR is lower right C is component

Table 20: FPGA Elements

Object	Assignable	Description	Location syntax
DLLDEL	Yes	Slave delay line for generating the desired delay in DDR/SPI4 applications. *ECP5, LatticeECP2/M, LatticeECP3, LIFMD/F, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	R<row> C<column>
DQS	Yes	Data strobe signal block that provides the required clock alignment for DDR memory interfaces. *Not available for LatticeSC/M*	LDQS<row> RDQS<row> TDQS<column> BDQS<column> where: L is left, R is right T is top, B is bottom.
DQSDLL	Yes	Delay-locked loop block for the data strobe signal. The DLL generates a 90-degree phase shift required for the DQS signal in DDR memory interfaces. *LatticeECP2/3, LatticeXP2, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	LDQSDLL, RDQSDLL where: L is left, R is right
DQSDLLTEST	No	Delay-locked loop for data strobe signal test. Used for internal hardware testing only.	
EBR	Yes	Embedded Block RAM, a large dedicated fast memory block that can be configured as ROM or RAM.	EBR_R<row> C<column> Example: EBR_R27C56
ECLKBRIDGECS	Yes	Edge clock bridge clock select, a high-speed clock bridge that takes the high-speed edge clock sources from the top and bottom. *ECP5, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	ECLKBRIDGECS0, ECLKBRIDGECS1
ECLKSYNC	Yes	Edge clock synchronization block for DDR memory. *ECP5, LatticeECP3, LIFMD/F, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	LECLKSYNC1, LECLKSYNC2, TECLKSYNC1, TECLKSYNC2, RECLKSYNC1, RECLKSYNC2, BECLKSYNC1, BECLKSYNC2 WHERE: L is left, R is right, T is top, B is bottom
EFB	Yes	Embedded function block for a hard IP. For the MachXO2 device, it contains a SPI, two I2Cs, and a timer/counter peripheral. The block is connected to a WISHBONE bus in slave mode. *MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	EFB

Table 20: FPGA Elements

Object	Assignable	Description	Location syntax
FSLICE	Yes	One of four interconnected slices in a programmable fast functional unit (PFF). Each FSLICE is capable of logic, ripple, and ROM modes of operation but not RAM. FSLICE sites are displayed in white by default.	R<row> C<column>
IOL or IOLOGIC	Yes	Input and output logic blocks.	L<row>A|B R<row> A|B T<column> A|B B<column> A|B where: L is left, R is right, T is top, B is bottom
JTAG	N/A	Joint Test Action Group Controller. It provides the control and interconnect circuit used by the boundary scan function	JTAG
M[3:0]	Yes	Non-JTAG dedicated configuration mode select pads. *LatticeSC/M*	The configuration mode pads can be assigned as user input by pin name. Example: LOCATE COMP "USERINPUT1" SITE "K23";
MAC	Yes	Multiply-accumulate block. *LatticeECP/2/3, LatticeECP2M, LatticeSC/M*	R<row> C<column>
MACO	Yes	Mask array for cost optimization block, an ASIC block for implementing various functions. *LatticeSCM*	MACO blocks must be located as shown in the logical preference file provided for each Lattice MACO IP. Example: LOCATE COMP "spi4_256ch/s4dp/s4dp_hc" SITE "LLMACO0" ;
MULT	Yes	Multiplier digital signal processing block that implements a multiply with no addition or accumulator nodes. *ECP5, LatticeEC, LatticeECP/2/3*	MULT9|MULT18|MULT36 R<row> C<column>
PCNTR	Yes	Power Controller block. *MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	PCNTR

Table 20: FPGA Elements

Object	Assignable	Description	Location syntax
PCS	Yes, when not connecting to a system bus	Physical coding sublayer. When connected to a system bus, the interconnect to the system bus determines the location. *ECP5, LatticeSC/M, LatticeECP3*	If not connecting to a system bus, the PCS should be assigned as described in the PCS application note. Example: LOCATE COMP "pcs_instance_name_in_hierarchy" SITE "PCS36000";
PERREG	Yes	Persistent User Register. Stores information from 16-bit registers and keeps the data intact during reconfiguration. *LatticeECP3*	PERREG
PICTEST	No	Programmable interface cell test, used for internal hardware testing only.	
PIO (P)	Yes	Programmable input-output block, often accompanied by an IOL or IOLOGIC block, that provide high-speed I/O registers and buffering to support a variety of signal interface standards. Two PIOs connected to sysIO buffers are the physical blocks of programmable interface cells (PICs).	L<row>A|B|C|D R<row> A|B|C|D T<column> A|B|C|D B<column> A|B|C|D where: L is left, R is right, T is top, B is bottom **Note**: Generally, PIOs should be located by their external pin names. For example, AK47
PLL	Yes	Phase-locked loop, a block with a voltage- or current-driven oscillator that is constantly adjusted to match in phase and therefore lock on the frequency of an input signal.	R<row> C<column> For LatticeSC and LatticeSCM devices: PLL_[UL][UR][LL][LR]C[AB] Example: PLL_LLCA where: UL is upper left, UR is upper right, LL is lower left, LR is lower right C is component
PLLREFCS	Yes	PLL reference clock select, which supports dynamic reference clock switching. *ECP5, LIFMF, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	LPLLREFCS (for 1k and above) RPLLREFCS (for 4k and above)
PRADD	Yes	Pre Adder for DSP block *ECP5*	R<row> C<column>

Table 20: FPGA Elements

Object	Assignable	Description	Location syntax
RESETN	Yes	Non-JTAG dedicated configuration res et input pad. RESETN is assigned to the specific input shown in the device's data sheet. *LatticeSC/M*	Example: LOCATE COMP "USERINPUTRESETBAR" SITE "H5";
SED	Yes	Soft error detect, an IP that checks for changes in the state of a configuration SRAM bit due to colliding alpha or neutron particles. The soft error detect IP can be set to automatically reconfigure the device on error. *ECP5, LatticeECP2/3, LatticeXP2, MachXO2, MachXO3D, MachXO3L, Platform Manager 2*	SED
SLICE	Yes	One of four interconnected slices in a programmable functional unit (PFU). Each SLICE is capable of logic, ripple, ROM, and RAM modes of operation. By default, SLICE sites are displayed in lavender.	R<row>C<column> A|B|C|D
SPLL	Yes	Simple phase-locked loop.	R<row>C<column>
STF	Yes	Store to flash block for user flash module (UFM) operations. *LatticeXP2*	STF
TSALL	Yes	Tri-state all, an element that puts all programmable input cells on the device in a tri-state condition. *MachXO, MachXO2,, MachXO3D, MachXO3L, Platform Manager, Platform Manager 2*	TSALL

Physical View

Physical View provides a read-only detailed layout of your design that includes switch boxes and physical wire connections. Routed connections are displayed as Manhattan-style lines, and unrouted connections are displayed as flylines. As you move your mouse slowly over the layout, the name and location of each REGION, group, component, port, net, and site are displayed as tool tips.

Displaying Components and Connections The Physical View toolbar allows you to select the types of elements that will be displayed on the layout.

 Displays local wires, those that expand across several blocks.

 Displays long wires, those that expand over a large area of the chip.

 Displays pin wires, short wires that are used to connect two or more net pins.

 Displays switch boxes, switching matrix resources located at the junctions of horizontal and vertical local lines.

 Displays switches, contact points employed for signal switching between resources on the chip.

 Displays sites, unused or vacant blocks to which logic can be assigned.

 Displays components, blocks that have assigned logic.

 Displays routes, physical connections between resources on the chip after the design has been routed.

 Displays unrouted nets, those that PAR failed to route or those that could not be routed because a site associated with a net was unlocked by the user.

 Displays the latest timing path selected in Timing Analysis View.

Displaying UGROUPs and REGIONs The Physical View toolbar also enables you to display or hide UGROUPs and REGIONs that exist in memory or that have been saved to the logical preference file.

 Display REGIONs – shows the bounding boxes of REGIONs.

 Display UGROUPs – shows the bounding boxes of UGROUPs.

Highlighting Elements Use the View menu's Highlight/Unhighlight commands to highlight selected nets or sites or to remove the highlighting.

Viewing Highlighted Items, Prohibited Sites, and Labels Use the View menu's Layer commands to display highlighted items, prohibited sites, and the textual labels of sites and components.

Note

When the text display has been turned on, a label will sometimes overlap the edge of the symbol or even appear aligned with the symbol below it. However, when you hold the mouse pointer over the symbol, the correct label will appear in the tool tip.

Displaying Partitions If your design includes partitions for the incremental design flow, you can display them in Physical View. See "Floorplan View and Physical View Partition Support" on page 674 for details.

See Also

- "Viewing Logic Details of Components" on page 448
- "Auto Cross-probing Between Floorplan, Physical, and Logic Block Views" on page 453
- "Using Incremental Design Flow" on page 651
- "Using the Zoom and Pan Tools" on page 454

Logic Block View

Logic Block View enables you to examine logic details of one or more placed and routed components. It provides either a schematic or tabular view, depending on the type of component selected.

Schematic views of PIO and PFU/PFF components can be accessed from NCD View, Floorplan View, and Physical View. Tabular views of PLL, EBR, and DSP blocks can be accessed from Floorplan View and Physical View.

See Also ▶"Viewing Logic Details of Components" on page 448

- "Auto Cross-probing Between Floorplan, Physical, and Logic Block Views" on page 453
- "Using the Zoom and Pan Tools" on page 454

Opening Logic Block View

You can open a schematic Logic Block View from Floorplan View, Physical View, or NCD View. Schematic views are available for PFU/PFF slice components or PIO components. For tabular views of EBR, PLL, and DSP components, you must use either Floorplan View or Physical View.

To open a Logic Block View from Physical View or Floorplan View:

1. Using the zoom tool, enlarge the area that contains the components you want to examine; or cross-probe to a single component from Netlist View or NCD View.
2. Select one or more components, right-click, and choose **Logic Block View** from the pop-up menu. Alternatively, double-click a single component to open it in Logic Block View.

To open a schematic Logic Block View from NCD View:

1. Select the Instances folder in NCD View.
2. From the Instances design tree, select one or more instances that you want to examine.
3. Right-click the selected instances and choose **Logic Block View**.

The Logic Block View opens in the main window. If you selected multiple components, the multiple views will be arranged in tabs at the top of the main window.

See Also ▶"Changing the Maximum Number of Logic Block Views" on page 448

- "Exporting an Image of a Schematic Logic Block View" on page 449

Changing the Maximum Number of Logic Block Views

You can use the Options dialog box to change the number of Logic Block Views that can be open at the same time. The default maximum number is 15, but you can change it to a lower number or increase it up to 25.

To change the number of Logic Block Views that can be open at the same time:

1. Choose **Tools > Options**.
2. In the Options dialog box, select **Physical View**, and then select **General**.
3. In the right pane, enter the desired number in the text box for "Maximum number of Logic Block Views can be opened (1-25)," and click **OK**.

Viewing Logic Details of Components

Logic Block View provides either a schematic or a tabular view, depending on the type of component selected.

Viewing Schematic Details of I/Os and PFUs The schematic Logic Block View displays color-coded details for device elements IOB, IOL, and PFU/PFF.

To view more details of a schematic Logic Block View:

- Use the zoom buttons to enlarge the display.
- Double-click a LUT to view a list of look-up table equations. Alternatively, select a LUT and click the EQU button on the vertical toolbar.
- Hold the mouse pointer over schematic symbols to view information in tool tips.

Viewing Tabular Details of PLL, EBR, and DSP Blocks The Logic Block View presents a tabular version of details for sysCLOCK PLL, sysMEM EBR, and sysDSP blocks and illustrates the device element programming. It lists the properties and shows the features that are enabled and the values for each. It also lists all physical site pins associated with the DSP block and includes port names.

For more detailed information on the types of blocks displayed in the tabular Logic Block View, see the following references:

sysPLL ▶How to Design with sysCLOCK PLLs and DLLs

- TN1263, *ECP5 sysCLOCK PLL/DLL Design and Usage Guide*
- TN1049, *LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide*
- TN1103, *LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide*
- TN1178, *LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide*

- TN1089, *MachXO sysCLOCK Design and Usage Guide*
- TN1199, *MachXO2 sysCLOCK PLL Design and Usage Guide*
- TN1282, *MachXO3L sysCLOCK PLL Design and Usage Guide*
- TN1098, *LatticeSC sysCLOCK and PLL/DLL User's Guide*
- TN1126, *LatticeXP2 sysCLOCK PLL Design and Usage Guide*
- TN1304, *CrossLink sysCLOCK PLL/DLL Design and Usage Guide*

sysMEM EBR ▶How to Design with FPGA Memories

- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1092, *Memory Usage Guide for MachXO Devices*
- TN1201, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1306, *CrossLink Memory Usage Guide*

sysDSP ▶How to Design with sysDSP

Exporting an Image of a Schematic Logic Block View

The Export command for Logic Block View enables you to capture a bitmap image of an open schematic view.

To export a schematic Logic Block View image:

1. Open the Logic Block View that you want to save as a bitmap image.
2. Choose **File > Export** or click the Export button on the vertical toolbar.
3. In the Export Image File dialog box, select the location for the image, and then type a name for the image in the File name text box.
4. Click **Save**.

   A bitmap image of the schematic view is saved in the directory you specified.

Cross-Probing Between Views

Diamond's preference-editing views allow you to select an element in one view and quickly display the corresponding logical or physical element in a different view. For example, you can cross-probe a component in Floorplan View to view the logical elements in Netlist View; or you can cross-probe a signal in Spreadsheet View to examine its placement on the pin layout of Package View.

To cross-probe an element from one view to another:

1. Right-click an element in any view.
2. Choose **Show in**, and then choose the desired view from the pop-up menu.

The following tables show the cross-probing availability for each preference view.

Table 21: Cross-probing from Spreadsheet View

Preference Sheet	Assignment	Cross-probe to
Port Assignments	Pin	Package View Device View Netlist View NCD View Floorplan View
Pin Assignments	Signal	Package View
Group	Port Group	Package View
	Anchored UGROUP	Floorplan View
Misc	REGION	Floorplan View

Table 22: Cross-probing from Package View

Element	Cross-probe to
Assigned pin	Netlist View Spreadsheet View Floorplan View Device View
Site	Floorplan View Device View

Table 23: Cross-probing from Device View

Element	Cross-probe to
DQS, IOL, PFF, PFU, PLL/DLL, DCC/DCS, sysDSP, sysMEM	Floorplan View Physical View
Others: GSR, JTAG, Oscillator, etc.	Floorplan View Physical View
Assignable PIO Cell	Package View Floorplan View Physical View
Unassignable PIO Cell	Floorplan View Physical View
DDR Support with bonded DQS	Package View Floorplan View Physical View

Table 24: Cross-probing from Netlist View

Element	Cross-probe to
Assigned port	Package View
Instance	Floorplan View Physical View NCD View
Net	Floorplan View Physical View

Table 25: Cross-probing from NCD View

Element	Cross-probe to
IOL instance	Netlist View Floorplan View Physical View
PCS block	Netlist View Floorplan View Physical View
PFF slice	Netlist View Floorplan View Physical View

Table 25: Cross-probing from NCD View (Continued)

Element	Cross-probe to
PFU slice	Netlist View Floorplan View Physical View
User PIO	Netlist View Package View Floorplan View Physical View
PLL/DLL	Netlist View Floorplan View Physical View
sysDSP block	Netlist View Floorplan View Physical View
sysMEM block	Netlist View Floorplan View Physical View
Others: GSR, JTAG, Oscillator, etc.	Floorplan View Physical View
Net	Floorplan View Physical View

Table 26: Cross-probing from Floorplan View

Element	Cross-probe to
Assignable PIO site	Device View Physical View Package View
Placed PIO	Device View Netlist View NCD View Physical View Package View
Placed IOL	Device View Netlist View NCD View Physical View

Table 26: Cross-probing from Floorplan View (Continued)

Element	Cross-probe to
Any non-PIO site	Device View Physical View
Any non-PIO placed component **Note**: To cross-probe PFF or PFU components, select individual slices.	Device View Netlist View NCD View Physical View
Port	Physical View
UGROUP	Spreadsheet View
REGION	Spreadsheet View

Table 27: Cross-probing from Physical View

Element	Cross-probe to
Site	Device View Floorplan View
Placed component	Device View Netlist View NCD View Floorplan View
Net	NCD View Floorplan View

See Also ▶"Auto Cross-probing Between Floorplan, Physical, and Logic Block Views" on page 453

Auto Cross-probing Between Floorplan, Physical, and Logic Block Views

When both Floorplan View and Physical View are open, an item that you select in one of these views is automatically selected in the other. This auto cross-probing is especially useful for immediately examining connections in both views. Additionally, when you open the Logic Block View for a selected component, the component is selected in both Floorplan View and Physical View.

The auto cross-probing tool does not zoom in to selected items or change the focus of the targeted view, and it does not open the targeted view if it is not already open. But when both Floorplan View and Physical View are open, auto cross-probing enables you to immediately examine objects in the context of each type of layout.

For example:

- Select a component in Floorplan View. Switch to Physical View and click the Zoom To button to view the selected component.
- Select a net in Physical View. Switch to Floorplan View to examine the net in the larger component layout.
- Double-click a component in Floorplan View to open Logic Block View.
  - Select a net in Logic Block View.
  - Switch to Floorplan View to view the highlighted net in the larger component layout.
  - Switch to Physical View to view the highlighted net in the more detailed layout.

See Also "Cross-Probing Between Views" on page 450

Using the Zoom and Pan Tools

The Diamond toolbar provides zoom tools for the layout views: Package View, Floorplan View, Physical View, and Logic Block View. In addition, the vertical toolbar included with Package View and Floorplan View provides commands that work together with the zoom tools to help you navigate to different objects and areas of the layout. Also, both Physical View and Logic Block View provide an area zoom that is controlled through the left mouse button. All of the layout views enable you to zoom in and out using the mouse wheel and function key combinations, and to pan using the arrow keys.

Diamond Toolbar Zoom Commands The following commands are available on the Diamond toolbar and in the View menu for each layout view.

Zoom In – enlarges the view of the entire layout.

Zoom Out – reduces the view of the entire layout.

Zoom Fit – reduces or enlarges the entire layout so that it fits inside the window.

Zoom To – enlarges the size of one or more selected sites, components, or connections on the layout and fills the window with the selection. The Zoom To command is available for all layout views except Logic Block View.

Area Zoom with the Mouse in Physical and Logic Block Views In Physical View and Logic Block View, you can enlarge an area of the layout by holding the left mouse button and dragging to the right and downward. Afterwards, you can zoom out of the area by dragging your mouse upward and to the left.

Vertical Toolbar Commands for Package View and Floorplan View The Select, Pan, and Zoom Area commands are available from the vertical toolbar in Package View and Floorplan View. They are also available from the "Mouse Tools" section of the View menu. These commands are mutually exclusive,

meaning that only one of them can be active at a time. Each allows you, however, to continue using the Diamond toolbar zoom commands.

Select – enables you to select objects on the layout by clicking an individual object or by dragging around an area to select multiple objects. It also enables you to move an assignment to a different location by dragging it. After selecting objects, you can zoom in with the Zoom To command. The Select command is activated by default. When Select is active, the Pan and Zoom Area commands are disabled.

Pan – enables you to use the mouse pointer to drag the layout in any direction. This command is useful for viewing a hidden area on the layout after zooming in. When Pan is active, the Select and Zoom Area commands are disabled.

Zoom Area – enables you to immediately enlarge an area of the layout by dragging to the right and downward with your mouse. When Zoom Area is active, the Select and Pan commands are disabled.

Panning with the Mouse in Physical and Logic Block Views After zooming in to a section of the layout, in Physical View or Logic Block View, you can press and hold the mouse wheel to drag the layout in any direction.

Zooming with Function Key Shortcuts The following key combinations, available for all the layout views, enable you to instantly zoom in or out from your keyboard:

- Zoom In – Ctrl++
- Zoom Out – Ctrl+-

Zooming with the Mouse Wheel The mouse wheel gives you finer zoom control, enabling you to zoom in or out in small increments. While pressing the **Ctrl** key, move the mouse wheel forward to zoom in and backward to zoom out. The mouse wheel zoom is available for all the layout views.

Panning with the Arrow Keys After enlarging the layout with the zoom tools, you can use the keyboard arrow keys to pan horizontally or vertically to different areas of the layout. Panning with the arrow keys is available for all the layout views.

Searching for Design Elements

Diamond's preference views enable you to search for design elements—either through a Find dialog box or through Find and Filter text boxes within the view.

Device View, Netlist View, and NCD View each provide Find and Filter text boxes at the top of the window. Diamond highlights the found objects and enables you to navigate through them and cross-probe to other views.

Spreadsheet View, Floorplan View, and Physical View each provide a Find dialog box. See the topic "Working with Preference Sheets" on page 412 for

information about searching in Spreadsheet View. The Find dialog box in Floorplan View and Physical View enables you to search for components, nets, and sites.

To search for design elements in Device View, Netlist View, or NCD View:

1. Open the desired view.
2. The Find and Filter text boxes appear at the top of the window.
3. To narrow the scope of the search, use wildcards in the Filter text box.
   - Use the asterisk to match one or more characters.
   - Use the question mark to match any single character.

   The view displays only those items that contain the characters you entered.
4. To locate a single design element from the list, type the name in the Find text box and press **Enter**.

   The found element is brought to the top of the view and highlighted.
5. To navigate through a list of elements that contain the same characters, use wildcards in the Find text box, and then press **Enter**.

   The first element containing the characters you entered is brought to the top of the view and highlighted. Press Enter again to go to the next element in the list.

To search for components, nets, or sites in Floorplan View or Physical View:

1. Open Floorplan View or Physical View, and then choose **Edit > Find** or press **Ctrl+F**.
2. In the Find dialog box, type the name of the component, net, or site that you want to find. Use the asterisk and question mark characters to perform wildcard searches.
3. Select the type from the Find Type drop-down menu.
4. Select Match Whole Word and Match Case as desired, to narrow your search to a complete name or to match upper and lower case letters.
5. Click **Find Next** or **Select All**.

   The item or items are highlighted on the layout.

   Use the Find Next and Find Previous buttons to navigate from one found item to another. Each will be highlighted on the layout.

See Also ▶"Going to a Specific Site" on page 439

Using Drag-and-Drop to Make Assignments

Diamond's preference-editing views provide a convenient drag-and-drop feature for making assignments or editing them. This feature is available between Netlist View and Spreadsheet View, between Netlist View and Package View, and between Netlist View and Floorplan View. The drag-and-drop feature is also available within Package View and within Floorplan View. This feature allows you to perform the following actions:

- Drag signals from Netlist View to Package View to assign them to pin locations.
- Drag assigned signals within Package View to reassign them to other locations.
- Drag nets from Netlist View to the Route Priority sheet of Spreadsheet View to set routing priority.
- Drag registers from Netlist View to the Cell Mapping sheet of Spreadsheet View to specify them as input or output flip-flops.
- Drag EBR, DSP, PCS, PLL, and ASIC block instances from Netlist View to Floorplan View to set LOCATE preferences.
- Drag non-SLICE logical components within Floorplan View to reassign them to other locations.
- Drag UGROUPs and REGIONS within Floorplan View to reassign them to other locations.

Undoing and Redoing Assignments

Diamond's Undo and Redo commands allow you to reverse preference changes that were made in a selected preference view. You can make multiple assignments, undo them all, and then redo them as desired. When all of your manual assignments have been undone or redone, the Undo or Redo command becomes dim. When you save your preference changes, the undo and redo commands are disabled.

Note

The Undo command will not remove the in-memory preference change indicators: the asterisk from a preference view tab or the "Preferences Modified" from the status bar. To remove these indicators after all assignments have been undone, use the Save command.

The commands are available from the Edit menu and from the Undo and Redo buttons on the Diamond toolbar.

The standard Windows keyword shortcuts are also available:

- Press **Ctrl+Z** to undo an assignment.
- Press **Ctry+Y** to redo an assignment.

Spreadsheet View Undo and Redo commands are available after design translation to reverse a preference change that was made in Spreadsheet View or pin assignments made in Netlist View. The commands are also available after Place & Route to reverse changes made in NCD View.

Package View Undo and Redo commands are available after design translation to reverse the following actions that were performed in Package View: signal re-assignments, unlocking of pin assignments, and prohibiting of pin sites.

Device View Undo and Redo commands are available after design translation to reverse any PROHIBIT preferences that were made in Device View.

Netlist View Undo and Redo commands are available after design translation to reverse any unlocking of assigned pins that was done in Netlist View.

Floorplan View Undo and Redo commands are available after design translation to reverse component, group, or REGION assignments and PROHIBIT preferences that were made in Floorplan View.

See Also ▶"Previewing Preferences Before Saving" on page 547

Printing a Preference View

You can print any preference sheet or any layout view. Diamond provides a preview for each view, enabling you to select options before printing.

Printing a Preference Sheet Any of the preference sheets can be printed. Categories, such as sysCONFIG, or groups are automatically expanded for the purpose of printing.

To print a preference sheet:

1. In Spreadsheet View, select the preference sheet that you want to print.
2. Choose **File > Print Preview**.
3. Select from the toolbar options at the bottom, and then click **Print**.

Printing a Layout View You can print a copy of the layout from Floorplan View, Package View, Physical View, or Logic Block View.

To print a layout:

1. Select the layout view that you want to print.
2. Choose **File > Print Preview**.
3. Select from the toolbar options at the top, and then click the **Print** button.

Setting Preferences

Preferences can be defined in the logical preference file (.lpf) at any stage of the design flow, but they are typically defined after synthesis and translation. You can use Diamond's preference-editing views or a text editor. Afterwards, the mapping process adds the changes to the physical preference (.prf) file and updates the native circuit description (.ncd).

Note

If you have a large design in the 100K LUT range and the logical preference file contains a large number of preferences (approximately five thousand), you might encounter the "out of memory" system error. See "Using Wildcard Expressions in Preferences" on page 533 to see how to reduce the number of preferences.

Diamond's preference-editing views allow you to specify many FPGA preferences, saving you the time it normally takes to write out logical preference syntax manually. These views include Spreadsheet View, Package View, Device View, Netlist View, NCD View, and Floorplan View.

Diamond implements simple error checking to ensure that the user assignments are applicable to the selected FPGA device and that there are no conflicting assignments. If the user preferences conflict with the selected device, the software will display these preferences in red. For example, if you change the device type after specifying some pin assignments, the non-applicable pin assignments will be displayed in red. You can delete these preferences within Diamond's preference views or within the .lpf file.

The validity of all the nets and ports are also checked against the logical native generic database (.ngd) file. If the .lpf file contains any of the physical naming in a net or a cell, it will result in an error condition. Physical names cannot be validated in the pre-map stage.

After the Translate Design process, you can view and edit any preferences that were passed from the HDL as attributes. When you edit these preferences or add new ones, you are writing changes in the form of logical preferences directly into the .lpf file. These preferences will be written to the physical preference file (.prf) by the mapping process.

Adding a Logical Preference File to a Project

When you create a new project in Diamond, a logical preference file (.lpf) is automatically generated and assigned the same name as the FPGA project. All changes that you make to logical constraints will be saved to this file until you create a new logical preference file or add an existing one.

Adding a new or existing .lpf file allows you to experiment with a different set of preferences for each implementation.

To add an existing .lpf file:

1. Choose **File > Add > Existing File**. Optionally, right-click the active implementation and choose **Add > Existing File**.
2. In the Add Existing File dialog box, do the following:
   a. Choose **Constraint Files (*.lpf)** from the "Files of type" drop-down menu.
   b. If you want to add the file to the implementation directory, select the option **Copy file to Implementation's Source directory**.
   c. Navigate to the .lpf file that you want to use and click **Add**.

   The selected .lpf file is added to the Constraint Files folder. If the file came from a directory outside your current project and you did not select the option to copy the file to the implementation directory, the relative path will be shown.

To create a new .lpf file and add it to the project:

1. Choose **File > New > File** or use the keyboard shortcut **Ctrl+N**. Optionally, right-click the active implementation and choose **Add > New File**.
2. In the New File dialog box, under Categories, select **Source Files**.
3. In the Source Files pane, select **Preference Files**.
4. In the Name box, type a new file name and select the **lpf** extension from the Ext drop-down list.
5. Accept the current directory for the new .lpf file or use the Browse button to select a different directory.
6. Select the **Add to Implementation** option to add the new file to the active implementation. Selecting this option also adds the file to the LPF Constraint Files folder in the File List view.
7. Click **New**.

   The new .lpf file opens in the Source Editor window. The file name appears in the Constraints Files list for your current implementation.

   Note

   If you did not select the Add to Implementation option, the file will not appear in the Constraint Files folder. To add it, use the procedure for adding an existing preference file.

See Also

- "Activating a Preference File" on page 461
- "Saving Preferences to a Different .lpf File" on page 547
- "Integrated Synthesis" on page 583

Activating a Preference File

When your project's Constraint Files folder contains more than one .lpf file, you can select one of them to serve as the active preference file for the current implementation.

To activate a preference file:

1. Select the .lpf file in the Constraint Files folder.
2. Right-click the selected file and choose **Set as Active Preference File**.

   If there are modified preferences or file changes in memory for your currently open project, the "Save Modified Files" dialog box will appear. If this happens, either click OK to save all files or clear the check marks in front of any files that you do not want to save and click OK.

The newly activated .lpf file now appears in bold type and will be used in processing the current implementation. All open preference views are refreshed and display any preferences contained in the newly activated file.

Setting Global Preferences

The Global sheet of Spreadsheet View enables you to set constraints that affect the entire design, such as voltage, temperature, global BLOCK preferences, and configuration preferences. The list of these preferences varies by device family.

To set global preferences:

1. In Spreadsheet View, select the **Global Preferences** tab.
2. To edit the value for a global or sysCONFIG preference, do one of the following:
   - Double-click the value and enter a new one.

     The table cell changes format. If the cell changes to a text box, type the new value. If it changes to a box with a drop-down menu, choose the value from the list.
   - Right-click the preference value cell and select the desired value from the pop-up menu. For a text entry, choose **Edit Cell**, and then type the new value.
3. To return to the default value, highlight the value and press the **Delete** key or right-click the value and choose **Clear**.

For ECP5, MachXO2, MachXO3D, MachXO3L, and Platform Manager 2 devices, the Global Preferences sheet allows you to specify the VCCIO voltage setting for each bank. The VCCIO setting will then appear in the read-only BANK_VCC column of the Port Assignments sheet for any signal or bank assignments. It will also appear in the BANK_VCC column of the Pin Assignments sheet. For other devices, the BANK_VCC preference can be set manually in the logical preference file (.lpf).

For LatticeSC/M devices, the Global Preferences sheet allows you to specify voltage derating preferences for VCC, VCC12P, VCCAUX, and VCCIO. For other devices, voltage derating preferences can be set manually in the .lpf file.

See Also ▶Global Preferences

- TN1260, *ECP5 sysCOFIG Usage Guide*
- TN1053, *LatticeEC/ECP sysCONFIG Usage Guide*
- TN1108, *LatticeECP2/M sysCONFIG Usage Guide*
- TN1169, *LatticeECP3 sysCONFIG Usage Guide*
- TN1080, *LatticeSC sysCONFIG Usage Guide*
- TN1082, *LatticeXP sysCONFIG Usage Guide*
- TN1141, *LatticeXP2 sysCONFIG Usage Guide*
- TN1204, *MachXO2 Programming and Configuration User Guide*
- TN1279, *MachXO3L Programming and Configuration Usage Guide*
- TN1303, *CrossLink Programming and Configuration Usage Guide*
- FPGA-TN-02069 - *Programming and Config Usage Guide*

Setting Temperature and Voltage Derating

The following tables show the min-max junction temperature range and the supported voltages for derating for each Lattice device family. The nominal values are specified where applicable. Lattice Diamond will use these values for DRC checks. Except for LatticeSC/M devices, voltage derating preferences are set manually in the .lpf file. They include VCC_DERATE, VCC1P2_DERATE, VCCAUX_DERATE, VCCIO_DERATE, and VREF_DERATE.

Note

The data ranges listed in the tables are derived from data collected during test conditions that might not match your unique operating conditions. You should compare your own data to ensure reliability. Also refer to relevant data sheets or application notes on the Lattice web site for updated information. Contact technical support for more information.

Table 28: LatticeECP/EC Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	100 C	N/A
VCC	1.14 V	1.26 V	1.2 V
VCCAUX	3.135 V	3.465 V	3.3 V

Table 28: LatticeECP/EC Temperature and Voltage Derating (Continued)

Symbol	Min.	Max.	Nominal
VCCIO	1.14 V	3.465 V	N/A
VCCREF	-0.5 V	3.75 V	N/A

Table 29: LatticeECP2/M Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATAURE	-40 C	100 C	N/A
VCC	1.14 V	1.26 V	1.2 V
VCCAUX	3.135 V	3.465 V	3.3 V
VCCIO	1.14 V	3.465 V	N/A
VCCREF	-0.5 V	3.75 V	N/A

Table 30: LatticeECP3 Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	100 C	N/A
VCC	1.14 V	1.26 V	1.2 V
VCCAUX	3.135 V	3.465 V	3.3 V
VCCIO	1.14 V	3.465 V	N/A
VCCREF			

Table 31: LatticeECP4 Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	100 C	N/A
VCC	1.14 V	1.26 V‘	1.2 V
VCCAUX	3.135 V	3.465 V	3.3 V
VCCIO	1.14 V	3.465 V	N/A
VCCREF			

Table 32: LatticeXP Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	100 C	N/A
VCC (1.2 V devices)	1.14 V	1.26 V	1.2 V
VCC (1.8 V, 2.5 V, 3.3 V devices)	1.71 V	3.465 V	2.5 V

Table 32: LatticeXP Temperature and Voltage Derating (Continued)

Symbol	Min.	Max.	Nominal
VCCAUX	3.135 V	3.465 V	3.3 V
VCCIO	1.14 V	3.465 V	N/A
VCCREF	-0.5 V	3.75 V	N/A

Table 33: LatticeXP2 Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	100 C	N/A
VCC	1.14 V	1.26 V	1.2 V
VCCAUX	3.135 V	3.465 V	3.3 V
VCCIO	1.14 V	3.465 V	N/A
VCCREF	-0.5 V	3.75 V	N/A

Table 34: LatticeSC/M Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	105 C	N/A
VCC	.95 V	1.26 V	1.2 V
VCC12	1.14 V	1.26 V	1.2 V
VCCAUX	2.375 V	2.625 V	2.5 V
VCCIO (Banks 1, 4, 5)	1.14 V	3.45 V	N/A
VCCIO (Banks 2, 3, 6, 7)	1.14 V	2.625 V	N/A
VCCREF	-0.5 V	3.6 V	N/A

Table 35: MachXO Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	100 C	N/A
VCC (1.2 V devices)	1.14 V	3.465 V	1.2 V
VCC (1.8 V, 2.5 V, 3.3 V devices)	1.71 V	3.465 V	N/A
VCCAUX	3.135 V	3.465 V	3.3 V

Table 35: MachXO Temperature and Voltage Derating (Continued)

Symbol	Min.	Max.	Nominal
VCCIO	1.14 V	3.465 V	N/A
VCCREF	-0.5 V	3.75 V	N/A

Table 36: MachXO2 Temperature and Voltage Derating

Symbol	Min.	Max.	Nominal
TEMPERATURE	-40 C	100 C	N/A
VCC (1.2 V devices)	1.14 V	1.26 V	1.2 V
VCC (2.5 V, 3.3V devices)	2.375 V	3.465 V	2.5 V or 3.3 V
VCCIO	1.14 V	3.465 V	N/A
VCCREF			

See Also ▶"Setting Preferences in a Text Editor" on page 527

- "TEMPERATURE" on page 1261
- "VOLTAGE" on page 1280
- "VCC_NOMINAL" on page 1276

Setting a Global Set/Reset Net Preference

The GSR_NET Preference dialog box allows you to specify a net as the input to the global set/reset (GSR) buffer inferred by the design mapper.

To set a Global Set/Reset Net preference:

1. In Spreadsheet View, select the Global tab.
2. Double-click **Global Set/Reset Net** to open the New GSR_NET Preference dialog box.

   Alternatively, right-click and choose **New GSR_NET**.
3. In the dialog box, select one or more nets from the Available Logical Nets list and click the **>** button to move the selection to the Selected Logical Nets list.

   You can use wildcards in the Filter text box to narrow the list of nets. To move all available nets to the Selected Logical Nets list, click the **>>** button.
4. Click **Add**.

Your selection of one or more nets is displayed on the Global preference sheet under “Global Set/Reset Net.” When you save the preference, each net is added to the .lpf file as a separate GSR_NET preference.

Assigning Signals

Diamond’s Spreadsheet View and Netlist View enable you to assign signals to pin locations. You can also assign signals by dragging them from Netlist View to pin locations on the Package View layout. Signals can be assigned or reassigned at any stage of the design flow.

Assigning Signals in Spreadsheet View

Spreadsheet View provides two methods for assigning signals to pin locations:

- Use the Port Assignments sheet to assign pins to listed signals. The Port Assignments sheet displays signals by port type and signal names.
- Use the Pin Assignments sheet to assign signals to listed pins. The Pin Assignments sheet lists pins by pin, pad name, bank, dual function, and polarity.

Both views enable you to make single or multiple assignments. Saved pin assignments are written to the logical preference file as LOCATE preferences.

Assigning Pins Using the Port Assignments Sheet For a single pin assignment, you can simply double-click the pin cell for a given signal and type the desired pin name. For multiple pin assignments, use the Assign Pins dialog box, as explained below.

To assign pin locations:

1. Select the pin cells, in the Pin column, for the signals that you want to assign.

   Note

   PCS-related signals for ECP5, LatticeECP2M, LatticeSC, and LatticeECP3 designs, are displayed in a dim font, indicating that they are read-only. For more information, refer to TN1261, *ECP5 SERDES/PCS Usage Guide*, TN1124, *LatticeECP2M SERDES/PCS Usage Guide*; TN1145, *LatticeSC flexiPCS/ SERDES Design Guide*; and TN1176, *LatticeECP3 SERDES/PCS Usage Guide*.

   To select multiple pin cells, do one of the following:

   - Drag down the pin column or use the Ctrl+click or Shift+click combination.
   - For a bus, open Netlist View and detach it. Select the bus from the Ports list of Netlist View and drag it to the Port Assignments Sheet of Spreadsheet View. All the rows for the bus signal names become highlighted.

2. Right-click the selected cells in the Pin column and choose **Assign Pins**.

   In the Assign Pins dialog box, the signals you selected are displayed in boldface in the left pane. In the pin list next to it, the same number of pins are displayed in bold at the top of the list. The software automatically interprets the design files and displays the pin filters that should be enabled by default. For example, if you selected an LVDS25 type of port in Spreadsheet View, the polarity section on the right will show the True P-side filter as enabled. If you selected a BLVDS type of port, it will show the Emulated P-side filter as enabled.

3. Clear or select the desired options under Pin Types to narrow or expand the list of available pins, and then select **Auto Sort** or **Manual Sort**.

   - If Auto Sort is selected, select the desired sorting option under “Sort by,” and then select **Ascending** or **Descending** order.
   - If Manual Sort is selected, use one of the following methods to drag pins and place them in the desired order:
     - Click one pin to select it, and then drag it to the desired location.
     - Use the Shift key to select several adjacent pins at once and release the mouse button. Afterwards, drag the highlighted pins to the desired position.

4. From the pin list, select the pin location where you would like the assignment to begin.

   The dialog box highlights the selected pin and displays in bold type the sequence of pins following it—one for each signal.

   Note

   If you have used the “Incompatible Pins” dialog box to select a device for possible pin migration, the incompatible pins will be displayed in a dim font on the pin list. If you have disabled the incompatible pins, Diamond will assign the next available pin for each pin that has been disabled.

5. From the Assign Pin drop-down menu at the bottom left, select the desired Assign Pin option: every, every second, every third, or every fourth pin.

6. To check the validity of the selected pin assignments, click **Check Pins**.

   A message box appears that lists any assignment errors. If errors are reported, repeat Steps 4–6 until the pin check is free of errors.

7. Click **Assign Pins**.

Assigning Signals Using the Pin Assignments Sheet For a single assignment, you can simply double-click the Signal Name cell for a selected pin to open the Assign Signals dialog box. For multiple assignments, right-click the selected cells.

To assign signals:

1. In the Signal Name column, select one or more cells for the signals that you want to assign. You can drag down the column to select multiple signal cells or use the Ctrl+click or Shift+click combination.

   Note

   If you have used the “Incompatible Pins” dialog box to select a device for possible pin migration, the incompatible pins will be displayed in a dim font. If you have disabled the incompatible pins, Diamond will assign the next available pin for each pin that has been disabled.

2. Right-click the selected cells in the Signal Name column and choose **Assign Signals**.

   In the Assign Signals dialog box, the pins you selected are displayed in boldface in the left pane. In the signal list next to it, the same number of signals are displayed in bold at the top of the list.

3. Clear or select the desired options under Signal Types to narrow or expand the list of available signals, and then select **Auto Sort** or **Manual Sort**.
   - If Auto Sort is selected, select the desired sorting option under “Sort by.”
   - If Manual Sort is selected, use one of the following methods to drag signals and place them in the desired order:
     - Click one signal to select it, and then drag it to the desired location.
     - Use the Shift key to select several adjacent signals and release the mouse button. Afterwards, drag the highlighted signals to the desired position.
4. Select **Ascending** or **Descending** order.
5. From the signal list, select the signal where you would like the assignment to begin.

   The dialog box highlights the selected signal and displays in bold type the sequence of signals following it—one for each pin.

6. To check the validity of the signal assignments, click **Check Signals**.

   A message box appears that lists any assignment errors. If errors are reported, repeat Steps 6 and 7 until the signal check is free of errors.

7. Click **Assign Signals**.

See Also
- “Modifying Assignments in Spreadsheet View” on page 469
- “Color Coding of Cell Values” on page 411
- “Defining IOBUF Preferences” on page 477
- “Setting Preferences for Port Groups” on page 508
- “Back Annotating Assignments” on page 491
- “IO_TYPE” on page 1310

- TN1262, *ECP5 sysIO Usage Guide*
- TN1056, *LatticeECP/EC and LatticeXP sysIO Usage Guide*.
- TN1102, *LatticeECP2/M sysIO Usage Guide*
- TN1177, *LatticeECP3 sysIO Usage Guide*
- TN1088, *LatticeSC PURESPEED I/O Usage Guide*
- TN1136, *LatticeXP2 sysIO Usage Guide*
- TN1091, *MachXO sysIO Usage Guide*
- TN1202, *MachXO2 sysIO Usage Guide*
- TN1280, *MachXO3L sysIO Usage Guide*
- TN1305, *CrossLink sysI/O Usage Guide*
- FPGA-TN-02068 - *MachXO3D sysIO Usage Guide*

Modifying Assignments in Spreadsheet View

You can modify pin assignments in Spreadsheet View before or after placement and routing. After routing, you can also modify assignments that were made automatically by the Place & Route Process.

To modify assignments:

1. From the Port Assignments sheet or Pin Assignments sheet, select the assignments that you want to modify, right-click, and choose **Assign Pins** or **Assign Signals**.
2. Follow the same procedure for assigning pins or signals in Spreadsheet View.

To modify assignments made automatically by Place & Route:

1. Choose **View > Display IO Placement**.

   All the placed and routed assignments are displayed in parentheses, including user assignments. User assignments appear twice—once without parentheses and once in parentheses—and they are displayed in black font or the font color that has been designated for “Normal” in the options dialog box. Assignments that were made automatically by Place & Route appear only once. They are displayed in parentheses and are distinguished by blue font or the font color designated for “Default Values.”
2. Select the assignments made by Place & Route that you want to modify, right-click, and choose **Assign Pins** or **Assign Signals**.
3. Follow the same procedure for assigning pins or signals in Spreadsheet View.

   The assignments are now displayed in black font. Those that were assigned by Place & Route now appear in parentheses, preceded by the new assignments.

To remove assignments:

- Select the assigned pins or signals that you want to delete, and then press the **Delete** key. Optionally, right-click the selected assignments and choose **Clear**.

  Note

  Only user assignments can be deleted. Assignments made automatically by Place & Route can be modified but not deleted.

Assigning Pins Using Netlist View

Netlist View enables you to select the signals from the design tree that you want to assign.

To assign pins:

1. From the Netlist View design tree, expand the category of signals that you want to assign.
2. Select the signals that you want to assign, right-click, and choose **Assign Pins**.
3. Follow the procedure for assigning pins using the Port Assignments sheet.

See Also
- "Assigning Signals Using Package and Netlist Views" on page 471
- "Back Annotating Assignments" on page 491

Modifying Pin Assignments Using Netlist View

Netlist View enables you to select the assigned signals from the design tree that you want to reassign.

To modify pin assignments:

1. Select the assigned signals that you want to modify, right-click, and choose **Assign Pins**.
2. Follow the procedure for assigning pins using the Port Assignments sheet.

   Note

   You can also modify one or more pin assignments by dragging from Netlist View to a different location in Package View.

To remove pin assignments:

- Select the assigned signals that you want to remove, right-click, and choose **Unlock**.

Assigning Signals Using Package and Netlist Views

You can drag signals from Netlist View onto the Package View pin layout to assign them or reassign them.

To assign signals using Netlist and Package views:

1. Choose **Tools > Package View**.
2. Click the Show Pin Display Selection button on the toolbar.

   The Pin Display Selection dialog box, by default, shows that all pin types and banks are selected.
3. In the dialog box, clear the types of pins and banks that you do not want to appear on the layout.
4. Click **Close**.
5. To display the fly wires for differential pins, choose **View > Show Differential Pairs**.

   Note

   For IO_TYPEs that require differential pins, Diamond automatically prohibits you from assigning the complement of a differential pin that is already assigned.
6. Choose **Tools > Netlist View**. The Ports design tree should be displayed. If it is not, click the Ports button on the toolbar.
7. Do one of the following:

   - Click the Detach Tool button at the top right to open Netlist View in a separate window, and then position Netlist View alongside Package View.
   - Click the Split Tab Group button on the toolbar to separate Netlist View into a tab group, and then select Package View in the main tab group. Both views now appear side-by-side.
8. From the Netlist View window, expand the list of ports and select the desired signals or bus.

   Note

   PCS-related signals for ECP5, LatticeECP2M, LatticeSC, and LatticeECP3 designs, are displayed in a dim font, indicating that they are read-only. For more information, refer to TN1261, *ECP5 SERDES/PCS Usage Guide*, TN1124, *LatticeECP2M SERDES/PCS Usage Guide*; TN1145, *LatticeSC flexiPCS/ SERDES Design Guide*; and TN1176, *LatticeECP3 SERDES/PCS Usage Guide*.
9. Drag your selection to the desired location on the Package View layout and release the mouse button.

   Package View fills the specified pin locations with color and assigns multiple signals in the order they were selected. The assignments are also shown in Netlist View and in the Port Assignments sheet of Spreadsheet View. When you save the assignments, each is written to the .lpf file as a LOCATE preference.

Click the “Integrate All Tools” button  to integrate the detached Netlist View with the main window.

Click the “Merge Tab Group” button  to merge the split tab into the main tab group.

See Also ▶“Back Annotating Assignments” on page 491

Modifying Signal Assignments in Package View

Package View enables you to use drag-and-drop to move one or more signal assignments to unlocked pins. It also allows you to move a single signal assignment to a pin that is already locked and swap the two assignments.

To modify one or more assignments in Package View:

1. Select the device pins on the layout that contain the assignments you want to modify.

   To select only one pin, click it once so that it is highlighted. Use the Ctrl key or the Shift key to select more than one pin, or drag your mouse around multiple pins to select them.

2. Slowly move the mouse toward the desired unlocked pin location.

   As you begin to drag, the cursor displays a crossed-out circle symbol. The cursor changes to an arrow as you move it over valid pins. Keep dragging until you have reached the targeted location.

3. When the cursor displays the arrow symbol over the targeted location, release the mouse button.

   Note

   If you are modifying a single assignment, make sure that the pin is highlighted before you try to drag it. If you try to drag a pin before it is highlighted, you will activate the multiple selection tool.

To swap two assignments:

1. Select the single device pin that contains the assignment you want to swap.
2. Drag the selected signal to the desired locked pin location and release the mouse button.
3. In the dialog box click **Swap assignments**.

To remove assignments in Package View:

1. Select the pins that contain the assignments you want to remove.
2. Right-click the selected pins and choose **Unlock**.

See Also ▶“Modifying Pin Assignments Using Netlist View” on page 470

▶ “Assigning Signals in Spreadsheet View” on page 466

- "Modifying Assignments in Spreadsheet View" on page 469
- "Migrating Pin Assignments" on page 484
- "Selecting a Device for Possible Pin Migration" on page 485
- "Viewing Incompatible Pins" on page 486

Modifying Signal Assignments in Floorplan View

In Floorplan View, you can modify a signal assignment by dragging it from one PIO to another. You can modify an assignment that originated from the logical preference file (.lpf) or one that was made automatically by Place & Route (PAR).

To modify a signal assignment in Floorplan View:

1. Make sure that the "Display Placement" button  is turned on if you will be modifying a PAR-assigned I/O. If you will be modifying an I/O assignment from the .lpf file, make sure that the "Display Placement Preferences" button  is turned on.
2. In Floorplan view, zoom in so that you can easily view the placed PIO component that you want to reassign and the targeted location.

   PIO components that were assigned automatically by PAR are displayed with a solid color fill. PIO components whose assignments originated from the .lpf file are displayed with a color fill and a cross-hatch overlay.
3. Drag from the assigned site to the targeted location and release the mouse button.

   The previous location is now displayed with a fill and no cross-hatch, and the targeted location is displayed with the cross-hatch and no fill. The targeted location will appear in the .lpf as a LOCATE preference when you save the change.

See Also

- "How Assignments are Displayed in Floorplan View" on page 435
- "Setting LOCATE Preferences in Floorplan View" on page 488

Setting Vref Locations

Lattice FPGA devices commonly provide two input pins per bank that can serve as on-chip voltage references (Vref) for the I/O types that require them. Spreadsheet View enables you to name and assign the specific voltage reference pins for these I/O types. If you do not assign Vrefs for I/O types that require them, the software will automatically create them for you. However, automatically generated Vrefs will not be added to the .lpf file if you decide to back annotate assignments.

After you have specified a Vref location, you can associate the Vref pin with a port group or with single signals. All Vrefs that you define are written to the logical preference file as LOCATE preferences.

Note

The Vref preference is not available for MachXO and Platform Manager devices.

To set Vref locations:

1. In Spreadsheet View, click the Vref Locations button  on the toolbar to open the Create New VREF dialog box.

   Alternatively, select the Misc Preferences sheet and double-click **VREF LOCATE** in the Preference Name column.
2. In the dialog box, under VREF Name, type a new VREF string.
3. If available, select a voltage rail from the Rail drop-down list.

   Note

   The Rail option is available and required for LatticeSC/M devices only.
4. Under SITE, select the desired pin location.
5. Click **Add**.

   The Misc Preferences sheet opens, displaying the list of Vrefs in the VREF LOCATE list.
6. To add more Vrefs, repeat steps 1-5.

When you save the design, Vref assignments appear in the .lpf file as LOCATE preferences.

To modify a Vref location:

1. Open the Misc Preferences sheet.
2. Expand VREF LOCATE and double-click the Vref that you want to modify.

   Alternatively, right-click the Vref and choose **Modify**.
3. In the Edit VREF Property dialog box, select the desired site and click **Update**.

To delete a Vref:

1. In the Misc Preferences sheet, select the Vref you want to remove.
2. Press the Delete key or right-click and choose **Delete**.

For more details on Vref usage, refer to the sysIO Usage Guide for your targeted device family:

- TN1262, *ECP5 sysIO Usage Guide*
- TN1056, *LatticeECP/EC and LatticeXP sysIO Usage Guide*
- TN1102, *LatticeECP2/M sysIO Usage Guide*

- TN1177, *LatticeECP3 sysIO Usage Guide*
- TN1088, *LatticeSC PURESPEED I/O Usage Guide*
- TN1136, *LatticeXP2 sysIO Usage Guide*
- TN1091, *MachXO sysIO Usage Guide*
- TN1202, *MachXO2 sysIO Usage Guide*
- TN1280, *MachXO3L sysIO Usage Guide*
- TN1305, *CrossLink sysI/O Usage Guide*
- FPGA-TN-02068 - *MachXO3D sysIO Usage Guide*

See Also

- "Defining IOBUF Preferences" on page 477
- "Setting Preferences for Port Groups" on page 508
- "Back Annotating Assignments" on page 491
- "IO_TYPE" on page 1310

Setting Port, Net, and Cell Attributes

Spreadsheet View's Port Assignments Sheet, Clock Resource sheet, Route Priority sheet, Cell Mapping sheet, and Global Preferences sheet enable you to set attributes for ports, nets, and cells. They allow you to set I/O port attributes, assign net priorities, specify registers for flip-flops, and set junction temperature and voltage.

Defining I/O Defaults

Diamond allows you to set values for all ports in your design at once. There are two ways to do this: use the ALLPORTS preference, which is available in Spreadsheet View, or manually create a DEFINE PORT GROUP preference in the .lpf file to create a group that includes all ports. Each of these methods allows you set default values while at the same time retaining values for individual ports or port groups that are set with other preferences.

For example, in a MachXO2 design, you might want to change the default I/O type to LVCMOS33, which would allow you to select a drive strength of 24 for individual output ports. But you might want to maintain the Vref settings for a port group with I/O type SSTL18_I. When you change the default to LVCMOS33 for all of the ports, Diamond changes the I/O type to LVCMOS33 for ports that use the default; but it maintains the SSTL18_I I/O type for the port group that uses the Vref assignment.

For example, in a LatticeSC design, you might want to change the default I/O type to HSTL15_1, which would allow you to select an impedance setting of 50 for individual output ports. But you might want to maintain the impedance setting of 100 that you have already set for a port with I/O type LVCMOS25. When you change the default to HSTL15_1 for all of the ports, the software changes the I/O type to HSTL15_1 for ports that use the default; but it

maintains the LVCMOS25 I/O type and impedance of 100 for the output port that you have already set.

You should use care when defining I/O defaults, especially if your design includes I/O settings in the HDL. Any existing IOBUF attributes in the HDL will get overridden by the Map process when an ALLPORTS preference or an all-ports PORT GROUP is used in the .lpf file. Before you define I/O defaults, make sure that you move any existing HDL IOBUF assignments to the .lpf file.

Defining I/O Defaults Using ALLPORTS The ALLPORTS row in Spreadsheet View's Port Assignments sheet enables you to quickly define a default setting for I/O attributes. It can be especially useful for setting the I/O type default.

To define default settings Using ALLPORTS:

1. In Spreadsheet View, select the Port Assignments tab.
2. Double-click the desired attribute cell in the ALLPORTS row and select a value from the drop-down list. Optionally, right-click the cell and select a value from the pop-up list.

   The newly selected value is displayed in orange, or the font color designated for “Reference Value,” for all of the ports that do not have other values already assigned.

   Note

   The range of attribute values available with the ALLPORTS row reflect the intersection of valid values across all port types. To access all available values, use individual ports or port groups.

   You can return to the original default setting by selecting the attribute cell and pressing the **Delete** key.

   The following is an example of an ALLPORTS preference.

```
IOBUF ALLPORTS IO_TYPE=LVCMOS33 PULLMODE=UP;
```

Defining I/O Defaults Using DEFINE PORT GROUP The DEFINE PORT GROUP preference enables you to use a wildcard to include all ports in one group. Afterwards, you can use Spreadsheet View to set IOBUF values for the entire group.

To define I/O defaults using a port group:

1. In the logical preference file, create the port group using the following syntax with the “*” wildcard:

```
DEFINE PORT GROUP "<group_name>" "*" ;
```

After you have created the port group and saved it in the .lpf file, the group appears at the bottom of Spreadsheet View's Port Assignments sheet. When there are other port groups in the .lpf file, Diamond issues a warning for each port that exists in one of these other groups.

2. In Spreadsheet View's Port Assignments sheet, scroll down to the port group row and select the desired IOBUF values from the appropriate columns.

The following is an example of an all-ports group, with IOBUF settings:

```
DEFINE PORT GROUP "all_ports" "*" ;
IOBUF GROUP "all_ports" IO_TYPE=LVCMOS18 PULLMODE=DOWN ;
```

See Also ▶"IO_TYPE" on page 1310

▶ "Conflicting IOBUF Constraints" on page 398

Defining IOBUF Preferences

Spreadsheet View's Port Assignments sheet enables you to define IOBUF preferences for individual ports and port groups. IOBUF attributes include BANK, IO_TYPE, PULLMODE, SLEWRATE, and many others. The available attributes will vary, depending on the selected Lattice device. These modified attributes will be written as IOBUF preferences in the logical preference file (.lpf).

The IOBUF preference does not support the use of wildcards. But you can use wildcards with the DEFINE GROUP preference to create a port group. Afterwards, you can assign IOBUF preferences to the entire group of ports at once. Port groups are shown at the bottom of the Port Assignments sheet. See also "Defining I/O Defaults" on page 475.

To define IOBUF preferences:

1. In Spreadsheet View, select the Port Assignments sheet.
2. Double-click the cell in the attribute column that you want to modify for a given signal or port group and choose a value from the drop-down menu. Optionally, right-click the cell and choose a value from the pop-up menu.

   Note

   For some designs, the OPENDRAIN attribute cannot be set to ON in Spreadsheet View but must be set in the HDL. Refer to the sysIO Usage Guide for the device family.

3. Press **Enter**.

   The modified value now appears in black font in the preference sheet or in the font color designated for "Normal Value."

See Also ▶"IOBUF Attributes" on page 478

- "Back Annotating Assignments" on page 491
- TN1262, *ECP5 sysIO Usage Guide*
- TN1056, *LatticeECP/EC and LatticeXP sysIO Usage Guide*
- TN1102, *LatticeECP2/M sysIO Usage Guide*
- TN1177, *LatticeECP3 sysIO Usage Guide*
- TN1088, *LatticeSC PURESPEED I/O Usage Guide*
- TN1136, *LatticeXP2 sysIO Usage Guide*
- TN1091, *MachXO sysIO Usage Guide*
- TN1202, *MachXO2 sysIO Usage Guide*
- TN1280, *MachXO3L sysIO Usage Guide*
- TN1305, *CrossLink sysI/O Usage Guide*
- FPGA-TN-02068 - *MachXO3D sysIO Usage Guide*

IOBUF Attributes

The following table describes the available attributes in the Port Assignments sheet for defining IOBUF preferences.

The options that you can select for these attributes will depend on the combinations that are legal for each device type. See the references below the table under See Also.

Table 37:

Attribute	Supported Devices	Description
BANK VCCIO	All	Sets the VCCIO voltage level of the IO bank.
CLAMP	ECP5, MachXO2, MachXO3D, MachXO3L, Platform Manager 2	Turns on the non-PCI-compliant clamp for smaller MachXO2 and MachXO3L devices. For 1200, 2k, 4k, 7k, and 10k devices, provides the option of setting the PCI-compliant clamp in the bottom bank.
DIFFCURRENT	LatticeSC/M	Sets the output current level for differential signals. Settings include 2, 3P5, 4, 6.
DIFFDRIVE	ECP5, LatticeECP3, MachXO2, MachXO3D, MachXO3L, Platform Manager 2	Sets the differential current drive strength for the MINILVDS output standard. Differential outputs with different DIFFDRIVE settings cannot be placed in the same IO bank. Settings include 1.6, 1.65, 1.7, 1.75, 1.81, 1.87, 1.93, 2.0. **Note**: The DIFFDRIVE value must be enclosed in quotation marks in the logical preference file.
DIFFRESISTOR	ECP5, LatticeSC/M, LatticeECP3	Provides differential termination for differential I/O types. Settings include OFF, 100, 150, 200. OFF is the default.

Table 37:

Attribute	Supported Devices	Description
DRIVE	All	Specifies the drive strength for bidirectional and output buffers.
EQ_CAL	LatticeECP3	Provides equalization filtering for single-ended inputs on both true and complementary I/Os and for differential inputs on true I/Os. **Note**: Only the default 0 value is valid for EQ_CAL.
HYSTERESIS	ECP5, MachXO2, MachXO3D, MachXO3L, Platform Manager 2	Sets the amount of hysteresis for the PCI, LVTTL and LVCMOS input and bidirectional I/O standards.
IMPEDANCE	LatticeSC/M	Sets the on-chip programmable output impedance. Depending on the I/O type and other selected attributes, settings can include OFF, 25, 33, 50, 100. OFF is the default.
IO_TYPE	All	Sets the I/O standard for the signal or port group.
MULTDRIVE	LatticeECP3	Sets the drive strength for individual output buffers for the MINILVDS output standard. Differential outputs with different MULTDRIVE settings can be placed in the same IO bank. Settings include 1X, 2X, 3X, 4X. **Note**: The MULTDRIVE value must be enclosed in quotation marks in the logical preference file.
OPENDRAIN	All	ON - specifies open-drain for an output or bidirectional buffer. OFF is the default. **Note**: For some designs, the OPENDRAIN attribute cannot be set to ON in Spreadsheet View but must be set in the HDL. Refer to the sysIO Usage Guide for the device family.
PCICLAMP	LatticeECP/EC, LatticeECP2, LatticeECP3, LatticeXP, LatticeSC/M, MachXO 1200/2280	ON - turns on the programmable PCI clamp diode. OFF is the default.
PULLMODE	All	Specifies the pull mode option: UP - pull-up (default) DOWN - pull-down NONE - pull off KEEPER - bus keeper
PWRSAVE	LatticeSC/M	Reduces power dissipation on comparator types of input pins. ON - Turns on the power save capability. OFF is the default.

Table 37:

Attribute	Supported Devices	Description
REFCIRCUIT	LatticeSC/M	Specifies a reference circuit for an output or bidirectional buffer. Settings include OFF, INTERNAL, EXTERNAL. OFF is the default.
SLEWRATE	All	Sets the slew rate for output and bidirectional buffers. Options are SLOW or FAST. SLOW is the default except for LatticeECP3 devices.
TERMINATEGND	LatticeSC/M	Sets the on-chip parallel input and output termination to ground. Depending on the I/O type and other selected attributes, settings can include OFF, 50, 100, 120. OFF is the default.
TERMINATEVCCIO	LatticeSC/M	Sets the on-chip parallel input and output termination to VCCIO. Depending on the I/O type and other selected attributes, settings can include OFF, 50, 100, 120. OFF is the default.
TERMINATEVTT	LatticeSC/M, LatticeECP3	Sets the on-chip input parallel termination to VTT. Depending on the I/O type and other selected attributes, settings can include OFF, 60 , 75, 120, 150, 210. OFF is the default.
TERMINATION	ECP5	Sets the on-chip input parallel termination to VCCIO/2. This parallel termination is achieved using a programmable Thevenin termination scheme of 50/75/150 ohms to VCCIO/2. OFF is the default.
VCMT	LatticeSC/M	Establishes a common mode voltage for input buffer elements. Depending on the I/O type and other selected attributes, settings can include OFF, VCMT, VTT, DDR_II. OFF is the default.
VREF	All except MachXO and Platform Manager	Assigns a defined voltage reference for the signal or port group.

See Also
- "IO_TYPE" on page 1310
- "Defining IOBUF Preferences" on page 477
- TN1262, *ECP5 sysIO Usage Guide*
- TN1056, *LatticeECP/EC and LatticeXP sysIO Usage Guide*.
- TN1088, *LatticeSC PURESPEED I/O Usage Guide*
- TN1102, *LatticeECP2/M sysIO Usage Guide*
- TN1177, *LatticeECP3 sysIO Usage Guide*

- TN1136, *LatticeXP2 sysIO Usage Guide*
- TN1091, *MachXO sysIO Usage Guide*
- TN1202, *MachXO2 sysIO Usage Guide*
- TN1280, *MachXO3L sysIO Usage Guide*
- TN1305, *CrossLink sysI/O Usage Guide*
- FPGA-TN-02068 - *MachXO3D sysIO Usage Guide*

Setting Clock Preferences

The Clock Resource sheet enables you to apply a clock domain to the device's primary or secondary clock or prohibit the use of primary and secondary clock resources in routing the net. For LatticeECP2 and LatticeXP2, it allows you to use edge clock resources to minimize the delay from the clock sources to I/O registers. For LatticeECP3 devices, it also enables you to specify a defined clock REGION, instead of a quadrant, for a secondary clock.

For clock enable (CE) or local set reset (LSR) signals, a secondary clock can be assigned or prohibited. Clocks for a clock net can be prohibited with the Prohibit Both command. This command writes a PROHIBIT PRIMARY and a PROHIBIT SECONDARY preference to the logical preference file (.lpf).

For a primary clock, you can also specify DCS or PURE for routing the clock spine. DCS will route the clock spine using the dynamic clock resource; PURE will route the clock spine without using the DCS routing resource.

The settings that you specify are written to the logical preference file (.lpf) as USE PRIMARY, USE SECONDARY, PROHIBIT PRIMARY, PROHIBIT SECONDARY, USE EDGE, and PROHIBIT EDGE preferences.

To set clock preferences:

1. In Spreadsheet View, select the Clock Resource sheet.
2. In the Selection, Quadrant, and DCS/PURE columns, double-click the cell for a given signal and choose the desired setting from the drop-down list. Optionally, right-click the cell and choose the setting from the pop-up menu.

Note

When a clock REGION is specified for a secondary clock, the REGION <region_name> <region_clock_space> preference must precede the USE SECONDARY preference that references it in the logical preference file.

Setting Output Load

The OUTPUT LOAD preference enables you to modify loading on output buffers from the default 0 pFs for timing analysis. You can specify the output load in the Port Assignments sheet of Spreadsheet View.

To set output load:

1. In Spreadsheet View, select the Port Assignments tab.
2. Double-click the cell in the Outload column that you want to modify for a given output signal and type the desired output load value.

Setting Maximum Skew

The MAXSKEW preference sets a maximum signal skew between a driver and loads on a specific clock signal.

To set the maximum skew:

1. In Spreadsheet View, select the Port Assignments tab.
2. Double-click the cell in the MaxSkew column that you want to modify for a given signal, and type the desired maximum skew value.
3. To specify the maximum skew for clock load only, double-click the Clock Load Only cell and choose **True**.

Assigning Route Priorities

The PRIORITIZE preference enables you to assign a priority to a selected net. Values range from 0 (lowest priority) to 100 (highest). This preference is used by the placement and routing (PAR) process, which assigns long lines by net priority and routes higher-priority nets before routing lower-priority nets.

To assign route priorities:

1. In Spreadsheet View, select the **Route Priority** tab.

   The Route Priority sheet contains three column headings: Type, Name, and Prioritize.
2. Choose **Tools > Netlist View** and click the nets button from the Netlist View toolbar.
3. Do one the following:

   - Click the Detach Tool button at the top right to open Netlist View in a separate window, and then position Netlist View alongside Spreadsheet View.
   - Click the Split Tab Group button on the toolbar to separate Netlist View into a tab group, and then select Spreadsheet View in the main tab group. Both views now appear side-by-side.
4. In Netlist View, expand the Nets list in the design tree.
5. Use Ctrl+click or Shift+click to select the nets that you want to prioritize.
6. Drag the selected nets to the Route Priority preference sheet and release the mouse button.

   Each net is given the default priority of 3.

7. To change the net priority of a selected net, double-click the Prioritize cell, and either type the desired value or select the value from the up and down arrows.
8. To return any nets to the default value, right-click the Prioritize cells for the selected nets and choose **Clear** or press the Delete key.

Click the “Integrate All Tools” button to integrate the detached Netlist View with the main window.

Click the “Merge Tab Group” button to merge the split tab into the main tab group.

To remove nets from the Route Priority sheet:

- Select the entire rows of the nets you want to remove, right-click, and choose **Delete Selected Rows**.

Specifying Registers for Flip-Flops

The USE DIN CELL and USE DOUT CELL preferences enable you to specify registers in your design that are to be used as input or output flip-flops. For each of these preferences, the True setting moves registers into the I/Os, and the False setting moves registers out of the I/Os.

To specify registers for flip-flops:

1. In Spreadsheet View, select the **Cell Mapping** tab.
2. Choose **Tools > Netlist View** and click the instances button from the Netlist View toolbar.
3. Do one the following:
   - Click the Detach Tool button at the top right to open Netlist View in a separate window, and then position Netlist View alongside Spreadsheet View.
   - Click the Split Tab Group button on the toolbar to separate Netlist View into a tab group, and then select Spreadsheet View in the main tab group. Both views now appear side-by-side.
4. To add registers to the Cell Mapping sheet, expand the Registers folder in the design tree of Netlist View, and then select the registers that you want to specify as flip-flops.
5. Drag the selected registers to the Cell Mapping preference sheet and release the mouse button.
4. In the Din/Dout column for the selected flip-flop, choose **DIN** or **DOUT** and press **Enter**.

   The PIO Register cell is automatically changed to TRUE.
5. To change the PIO Register value, double-click the PIO Register cell and choose **FALSE**.

6. To undo the any of the values you have assigned, right-click the selected Prioritize cells and choose **Clear** or press the Delete key.

Click the “Integrate All Tools” button  to integrate the detached Netlist View with the main window.

Click the “Merge Tab Group” button  to merge the split tab into the main tab group.

To remove registers from the Cell Mapping sheet:

- Select the entire rows of the registers you want to remove, right-click, and choose **Delete Selected Rows**.

Specifying Voltage and Temperature

The Global preference sheet enables you to define the device core voltage and core temperature for any Lattice device. For LatticeSC/M devices, it also allows you to set other voltages for derating.

To specify voltage and temperature settings:

1. In Spreadsheet View, select the Global Preferences tab.
2. In the Preference Value column, double-click the cell for Junction Temperature and type a new value. Do the same for Voltage.
3. For LatticeSC/M devices, double-click the Preference Value cell for Derating and choose the desired value from the drop-down list.

Migrating Pin Assignments

Diamond provides pin compatibility information that helps you migrate pin assignments to a different device of the same family and package as the currently targeted device. This information enables you to save your current pinout while you explore other devices. It shows you which pins in your current device will be unavailable or have different functions when you switch devices, and it allows you to export the pin migration information to a pin layout file.

You use the “Incompatible Pins” dialog box to select one or more devices that are of the same family and package as your current device. Afterwards, you can view all incompatible pins in Spreadsheet View and Package View. The rows for incompatible pins are dimmed on the Pin Assignments sheet of Spreadsheet View. In Package View, a dark gray circle appears behind each incompatible pin square.

Pin compatibility information is available after the Translate Design process. It is available for all device families except Platform Manager.

Note

Pin migration is not available for LatticeSC/M devices.

See Also ▶"Selecting a Device for Possible Pin Migration" on page 485

- "Viewing Incompatible Pins" on page 486
- "Exporting Pin Migration Information" on page 488
- "Exporting a Pin Layout File" on page 564
- "Color Coding of Cell Values" on page 411

Selecting a Device for Possible Pin Migration

The Incompatible Pins dialog box allows you to select one or more devices for possible pin migration. Afterwards you can view the incompatible pins—those that would not be available or that would have a different function if you were to switch from the current device.

To select one or more devices for pin migration:

1. Open Spreadsheet View or Package View.
2. Choose **View > Show Incompatible Pin** or click the button on the toolbar.

   The Incompatible Pins dialog box lists all the devices that are of the same device family and package as the current device.
3. Select one or more of the devices from the Compatible Devices list, or select **All Devices**.

   If you select one device from the list, the incompatible pins will include those between the current device and the selected device.

   If you select more than one device or All Devices, the incompatible pins will include those between the current device and the combination of all selected devices.

   Note

   The LFE3-17EA, in a 484 package, may have a DQS grouping migration issue that requires design changes if DQS-DQ associations are used in the design. Refer to the ECP3 Data Sheet and the ECP3 pinout files for the specific devices to check for DQS grouping mismatches.
4. In the Check Areas section, select the types of incompatibility that you want included for the selected devices: pins whose type, configuration, LVDS/high speed, and bank numbers differ between the current device and the selected devices.
5. Optionally, select **Disable Incompatible Pins Assignment** to unlock any previously assigned pins that would now be incompatible.

6. Click **OK**.

   If you did not select the “Disable Incompatible Pin Assignments” option, all of the pins that are incompatible will be displayed in a dim font on the Pin Assignments sheet and with a dark gray circle in Package View.

   If you selected the “Disable Incompatible Pin Assignments” option and your design includes pin assignments that are incompatible with the selected devices, a dialog box will warn you that all assignments to incompatible pins will be released. Do one of the following:

   - Click **Yes** to continue.

     This releases the pin assignments that are now incompatible but retains all other pin assignments. All incompatible pins will be displayed in a dim font on the Pin Assignments sheet and with a dark circle in Package View. Names of previously assigned signals whose pins are now incompatible will not be displayed.

     There is no Undo for this operation. If the design has already been mapped, it will be initialized back to pre-map status when you save the change.

   - Click **No** if you decide that you do not want to release the previously assigned pins.

     This maintains all of your current assignments but displays none of the pins as incompatible. To display incompatible pins while maintaining the assignments, you must re-open the Incompatible Pins dialog box and select other devices.

See Also ▶“Viewing Incompatible Pins” on page 486

- “Prohibiting Incompatible Pins” on page 487
- “Spreadsheet View” on page 408
- “Package View” on page 426
- “Assigning Signals in Spreadsheet View” on page 466
- “Migrating Pin Assignments” on page 484
- “Exporting Pin Migration Information” on page 488
- “Exporting a Pin Layout File” on page 564
- “Color Coding of Cell Values” on page 411

Viewing Incompatible Pins

An incompatible pin is one that would not be available or would have a different function if you were to switch your design from the current device to a different device. After selecting one or more devices of the same family and

package for possible pin migration, you can view the incompatible pins in Spreadsheet View and Package View. In Package View, you can also view the tool tip that explains why a device pin is incompatible.

Note

None of the pin sites that are marked as incompatible can be assigned. If you need to use any of these sites for your design, you must return to the “Incompatible Pins” dialog box and adjust your device selections.

To view incompatible pins in Spreadsheet View:

- Select the Pin Assignments sheet and scroll through the list of pins.

  The rows for pins that are incompatible are dimmed.

  Note

  When you use the Assign Pins dialog box from the Port Assignments sheet, all incompatible pins in the dialog box are dimmed.

To view incompatible pins in Package View:

- In Package View, maximize the view or choose **View > Zoom Fit**.

  On the Package View layout, each incompatible pin is displayed with a dark gray circle behind the pin square.

- Hold your mouse pointer over an incompatible pin to view the tool tip.

  The explanation of why the pin is incompatible is listed at the end of the tool tip and will include one of the following:

  “[TYPE] is not compatible”

  “[HIGHSPEED] is not compatible”

  “[CFG] is not compatible”

  “[Bank Number] is not compatible”

See Also
- “Spreadsheet View” on page 408
- “Package View” on page 426
- Migrating Pin Assignments
- “Exporting Pin Migration Information” on page 488
- “Exporting a Pin Layout File” on page 564

Prohibiting Incompatible Pins

Both Spreadsheet View and Package View enable you to create a PROHIBIT preference for selected incompatible pins or all incompatible pins. Prohibiting incompatible pins will prevent you from assigning signals to them. It will also prevent the Map and Place & Route processes from using these incompatible pins. When you prohibit incompatible pins, none of the pins that have already been assigned will become unlocked.

To prohibit all incompatible pins:

1. In Spreadsheet View or Package View, select the devices for possible pin migration.
2. Click the **Prohibit Incompatible Pins** button on the toolbar.

   On the Pin Assignments sheet of Spreadsheet View, “Prohibit” appears in the signal name column for all prohibited incompatible pins.

   In Package View, all prohibited incompatible pins are highlighted in red.

To release all prohibited incompatible pins:

- Click the **Release Incompatible Pins** button on the toolbar.

To prohibit one or several incompatible pins:

- On the Pin Assignment sheet of Spreadsheet View or in Package View, select the incompatible pins that you want to prohibit. Right-click, and choose **Prohibit**.

To release one or several incompatible pins:

- On the Pin Assignment sheet of Spreadsheet View or in Package View, select the prohibited pins that you want to release. Right-click and choose **Release**.

See Also “Viewing Incompatible Pins” on page 486

- “Selecting a Device for Possible Pin Migration” on page 485

Exporting Pin Migration Information

Diamond allows you to include pin migration information in an exported pin layout file of your design. The exported pin layout file uses a delimiter-separator format, such as comma-separator value (.csv), which allows you to view the report in an external spreadsheet application. In addition to pin migration information, the pin layout file provides a report of available device pins, pin assignments, and IOBUF attributes. You can include pin migration information in the report by selecting the **Pin Migration** option in the Export Pin Layout File dialog box.

See “Exporting a Pin Layout File” on page 564

Setting LOCATE Preferences in Floorplan View

You can set LOCATE preferences in Floorplan View for I/O, EBR, PLL, DLL, and DSP components. There are three ways to make these assignments:

- Drag and drop selected elements from Netlist View to Floorplan View

- Use the “Set LOCATE Preference” command for placed components in Floorplan View.
- Drag an assigned component to a new location in Floorplan View.

When you save the change, Diamond saves each assignment with a LOCATE COMP preference in the logical preference file (.lpf).

Dragging Elements from Netlist View to Floorplan View You can drag and drop I/O, EBR, PLL, DLL, and DSP elements from Netlist View to Floorplan View at any stage of the design flow after design translation.

To set LOCATE preferences using Netlist View and Floorplan View:

1. Open both Netlist View and Floorplan View, and then position each view next to each other by doing one the following:
   - Select Netlist View and click the Detach Tool button at the top right to open it as a separate window. Do the same for Floorplan View, if desired, and then position both views next to each other.
   - Click the Split Tab Group button on the toolbar to separate Netlist View into a tab group, and then select Floorplan View in the main tab group. Both views now appear side-by-side.
2. In Floorplan View, zoom in as needed to the area where you want to make the assignment.
3. In Netlist View, do one of the following:
   - Click the Ports button , and select one or more ports for I/O assignments.
   - Click the Instances button and expand the appropriate folder. Select one or more instances for EBR, PLL, DLL, or DSP assignments.
4. Using the left mouse button, drag the selection to the desired sites on the Floorplan View layout, and release the mouse button.

   If you make an error, choose **Edit > Undo**.

   Netlist View displays the newly assigned locations, and each site is filled with color in Floorplan View. When you save the change, each assignment is written to the .lpf file as a LOCATE COMP preference. The placement will be retained in the next Place & Route process.

Click the “Integrate All Tools” button to integrate the detached Netlist View with the main window.

Click the “Merge Tab Group” button to merge the split tab into the main tab group.

Using the “Set LOCATE Preference” Command in Floorplan View After placement and routing, you can set LOCATE preferences directly on the Floorplan View layout. You can do this for one or more I/O, EBR, PLL, DLL, or DSP components that you want placed in the same location that was

assigned by Place & Route. This will ensures that the placement will be retained when you rerun PAR. When you save the changes, Diamond saves each instance with a LOCATE COMP preference in the logical preference file (.lpf).

To assign LOCATE preferences using Floorplan View:

1. In Floorplan View, use the zoom tools to enlarge the area that contains the instances that you want to reassign.
2. Select the assigned components, using one of the following methods:
   - Click a single component.
   - Use Ctrl+click to select more than one component.
   - Click a component in a row or column, and then hold the Shift key while you select a different component in the same column or row. Diamond highlights all components and sites within range of the two selected components.
   - Drag the mouse pointer around the area that contains the desired components. All components and sites within the area become highlighted.
3. To keep the locations assigned by Place & Route, right-click the selected components and choose **Set LOCATE Preference**.

   The components are now displayed with a cross-hatch overlaying the solid color fill.
4. Choose **File > Save**.

   Diamond saves each assignment as a LOCATE COMP preference in the .lpf file. The placement will be retained in the next Place & Route process.

Dragging an Assigned Component to a New Location in Floorplan View You can drag an assigned component to a new location in Floorplan View. You can do this for components that have already been assigned with a LOCATE preference or for those that were automatically assigned by Place & Route.

1. In Floorplan View, use the zoom tools to enlarge the area that contains the assigned components that you want to reassign.
2. Select one or more I/O, EBR, PLL, DLL, or DSP components, using one of the following methods:
   - Click a single component.
   - Use the Ctrl key to select more than one component.
   - Click a component in a row or column, and then hold the Shift key while you select a different component in the same column or row. Diamond highlights all components and sites within range of the two selected components.

- Drag the mouse pointer around the area that contains the desired components. All components and sites within the area become highlighted.

Note

DSP components can be dragged and dropped only after Place & Route.

3. Using the left mouse button, drag the selected components to the desired sites, and then release the mouse button.

   The targeted sites are now displayed with a cross-hatch overlay. The previously assigned locations are still displayed with a solid color fill. The components will be moved to the new location when you rerun Place & Route.

Back Annotating Assignments

To ensure that subsequent Place & Route (PAR) processes produce the same results, Diamond enables you to back annotate placement and configuration assignments to the logical preference file (.lpf), where they can be modified as needed. The back annotation process gathers the assignments produced by PAR from the physical design (.ncd) file, and it places them in memory. They can then be viewed in the Preference Preview before saving. When you save the assignments, Diamond writes the placement assignments to the .lpf file as LOCATE preferences, and it writes the configuration assignments as IOBUF preferences. The design is then returned to the pre-map stage.

To back annotate assignments:

1. Run the **Place & Route Design** process to obtain a physical design (.ncd) file for the design.
2. Open Package View, Spreadsheet View, or Floorplan View.
3. Choose **Design > Back Annotate Assignments**.
4. In the dialog box, select the types of assignments that you want to back annotate and click **OK**.

   Diamond gathers the assignments from the .ncd file and places them in memory.
5. To view the assignments before saving, choose **View >  Preference Preview**.
6. To write the assignments to the .lpf file, return to Package View, Spreadsheet View, or Floorplan View, and then choose **File > Save <file_name>.lpf**.

   Diamond back annotates the assignments to the active .lpf file and re-initializes the design to the pre-map stage.
7. Modify preferences as needed, and then rerun **Place & Route Design**.

See Also ▶"Setting LOCATE Preferences in Floorplan View" on page 488

- "Displaying I/O Placement Assignments in Spreadsheet View" on page 422
- "Displaying I/O Placement Assignments in Package View" on page 429
- "How Assignments are Displayed in Floorplan View" on page 435

Setting Timing Preferences

The first five buttons on Spreadsheet View's toolbar gives you quick access to timing preference dialog boxes. You can also access the dialog boxes from the Timing Preferences spreadsheet and from Spreadsheet View's Edit > Preference menu.

 adds a BLOCK preference.

 adds a PERIOD or FREQUENCY preference.

 adds an INPUT_SETUP or CLOCK_TO_OUT preference.

 adds a MULTICYCLE preference.

 adds a MAXDELAY preference.

Auto Generating Timing Preferences

If you do not define any timing preferences for your FPGA design, the software will automatically generate a few timing preferences when you run the Map TRACE process. It generates FREQUENCY preferences to set your fMAX for each clock network and writes these preferences to the physical preference file (.prf). This estimated maximum frequency drives the PAR process to provide better performance results. You can either accept these preferences or use them as the starting point to set your own timing preferences.

By default, the Auto Timing option is turned on in the Strategy dialog box for the Map Trace process and the Place & Route Trace process. To prevent the auto generation of timing preferences, change the Auto Timing setting to False for each of these processes.

Note

If you have specified even a single timing preference, the auto-generation will not happen.

When the Auto Timing option is turned on, the software checks the project's physical preference file (.prf) during the Map Design process to see if it contains any timing preference. If no timing preferences are found, the software will automatically generate fMAX preferences for each clock network and write them to the .prf file. These timing preferences represent the estimation of fMAX, which can drive the Place & Route process to get better performance.

The following example shows auto-generated preferences that have been added to the .prf file:

Figure 79:

```
// Section Autogen
BLOCK RESETPATHS ;
BLOCK ASYNCPATHS ;
FREQUENCY PORT "CLK" 346.741 MHz ;
FREQUENCY NET "CLK_int" 346.741 MHz ;
// End Section Autogen
```

When auto-generating timing preferences, the software identifies all clock sources at input ports and internal nets, which drive the clock pin of any internal register/RAM/DSP or I/O register in a given design. The fMAX preference is generated for each clock source input port or net. There is no preference generated for any multi-domain path crossing over different clock source input ports. For a gated clock such as a clock from an AND gate, both input ports to 2-input of an AND gate will be identified as clock source with Fmax preference based on MAP-module-delay result.

The auto-generated timing preferences are written to the .prf file. The logical preference file (.lpf) does not contain them. When you rerun the Map Design process, the software regenerates the .prf file from the preferences defined in the .lpf file and design source files, which means that the auto-generated preferences in the previous .prf file will be lost. If you want to retain these preferences, copy them into your .lpf file before you remap your design.

To get auto-generated timing preferences and adjust them based on your actual requirement, run the Map process without setting any timing preferences. The software will auto generate timing preferences and write them to the .prf file in the project directory. You can then open the .prf file in a text editor as a reference and edit the .lpf file accordingly. Otherwise, you can copy the timing preferences from the .prf file to the .lpf file and open Spreadsheet View to edit them. When you remap the design, the previous .prf file with auto-generated preferences will be overwritten.

Setting BLOCK Preferences

A BLOCK preference is useful for excluding specific nets or paths from timing analysis that are not relevant to the synchronous timing of your design. The Block Preference dialog box enables you to block one or more nets or buses and all paths through them or block a path to and from specified groups or objects. The BLOCK preferences are displayed in the Timing Preferences sheet of Spreadsheet View.

You can also set global preferences to block all paths of a certain type from timing analysis. See “Global Preferences” on page 384.

To block a path that goes through a specified pin of a named component, you can set a BLOCK COMP PIN preference in the .lpf file, using a text editor. See “Setting Preferences in a Text Editor” on page 527.

To open the BLOCK preference dialog box:

- In Spreadsheet View, click the Block Preference button on the toolbar.

  Alternatively, open the Timing Preferences sheet and double-click **BLOCK** in the Preference Name column.

To block a net: 1.In the Block Preference dialog box under Type, select the **Block Nets** option.

All the nets in your design appear in the Available Nets list. You can use the Filter box and wildcards to narrow the list of displayed nets or to create a single BLOCK preference for multiple nets.

2. Do one of the following:
   - Select the net that you want to block
   - Using wildcards, type a string (case-sensitive) in the Filter box to create a single BLOCK preference for multiple nets.
3. Click **OK**.

   The new preference is added to the BLOCK Preference list on the Timing Preference sheet.

To block a bus: 1.In the Block Preference dialog box under Type, select the **Block Buses** option.

All the buses in your design appear in the Available Buses list. You can use the Filter box and wildcards to narrow the list of displayed buses or to create a single BLOCK preference for multiple buses.

2. Do one of the following:
   - Select the bus that you want to block.
   - Using wildcards, type a string (case-sensitive) in the Filter box to create a single BLOCK preference for multiple buses.
3. Click **OK**.

   The new preference is added to the BLOCK Preference list on the Timing Preference sheet.

To block the path from one group to another: 1.In the Block Preference dialog box under Type, select the **Block Path for** option, and then select **Group** from the drop-down menu.

All the port, cell, and ASIC groups that have been defined in your design are displayed in the two lists. You can use the Filter box and wildcards to limit the number of groups displayed in each list. You cannot, however, create a single BLOCK preference from multiple groups.

2. In the “From User Groups” list, select the source group.
3. In the “To User Groups” list, select the destination group.
4. Click **OK**.

   The new preference is added to the BLOCK Preference list on the Timing Preferences sheet.

To block the path from one clock net, port, cell, or ASIC to another:

1. In the Block Preference dialog box under Type, select the **Block Path for** option, and then select the desired object type from the drop-down menu.

   Depending on your selection, all the clock nets, ports, cells, or ASICs in your design are displayed in the appropriate “From” and “To” lists. You can use the Filter box and wildcards to limit the number of objects displayed in each list. You cannot, however, create a single BLOCK preference from multiple objects.

2. In the “From” list, select a clock net, port, cell, or ASIC path source.
3. In the “To” list, select a clock net, port, cell, or ASIC path destination. For ASICs, also select an ASIC pin from the “From Pins” list and the “To Pins” list.
4. Clock **OK**.

   The new preference is added to the BLOCK Preference list on the Timing Preferences sheet.

Setting PERIOD and FREQUENCY Preferences

The PERIOD preference allows you to verify sequential data paths internal to the design. It identifies a clock period for all sequential output to sequential input pins clocked by the specified net. If no net name is given, the specific clock period will apply to all sequential input pins that do not have a specific clock period.

The FREQUENCY preference identifies the minimum operating frequency for all sequential output to sequential input pins clocked by the specified net. If no net name is given, the preference applies to all clock nets in the design that do not have a specific clock frequency. FREQUENCY works in the same way as PERIOD, except that time is specified in Hz instead of seconds.

You can set PERIOD and FREQUENCY preferences using the PERIOD/FREQUENCY Preference dialog box. These preferences are then displayed in the Timing Preferences sheet of Spreadsheet View.

To set a PERIOD preference: 1.In Spreadsheet View, do one of the following:

- Click the Period/Frequency button  on the toolbar to open the dialog box, and then select **PERIOD**.
- Open the Timing Preferences sheet and double-click **PERIOD** in the Preference Name column. When the dialog box opens, the PERIOD preference type will already be selected.

2. Under Second Type, do one of the following:

   - Select **None** if you want to specify a clock period that applies to all sequential input pins that do not have a specific clock period.
   - Select **Clock Net** if you want to specify a clock period for all sequential output to sequential input pins clocked by a specified net.

- Select **Clock Port** if you want to assign a clock period to the specified top-level clock signal port.

3. If you have chosen **Clock Net** or **Clock Port** in step 2, select one or more objects in the Available Clock Nets or Available Clock Ports list.

   For nets, you can use the Available Clock Nets list Filter and wildcard notation to create a single PERIOD preference for multiple clock nets. Using wildcard notation, type a string (case-sensitive) to limit the list to the clock nets you want to include, and then click Go.

   See "Using Filters and Wildcards" on page 424.

   For ports, you can use the filter to limit the list of ports displayed. You must then select a clock port from the list.

4. Enter a value in nanoseconds for Time, and do the same for Hold Margin.
5. In the PAR_ADJ box, type the desired value. The PAR_ADJ option is available when **Clock Net** or **Clock Port** is selected under Second Type.

   Note

   PAR_ADJ allows you to loosen requirements for place-and-route results while tightening up the requirements reported by the TRACE static timing analysis tool. The variance in required timing values on PAR and TRACE enables you to experiment more efficiently with the strategy of purposely over-constraining your design to determine your best constraint settings. For example, if FREQUENCY is set to 250MHz and you set PAR_ADJ to 20, the software will use the simple addition formula of 250 + 20 = 270MHz to direct PAR while running TRACE on 250MHz.

6. In the CLOCK_JITTER (p-p) box, which is available for a clock port only, type a value in nanoseconds for peak-to-peak jitter on the incoming clock.

   Trace will use half of this peak-to-peak jitter value for all analysis.

7. Click **OK**.

   The Timing Preferences sheet displays the new preference in the PERIOD list.

To Set a FREQUENCY Preference 1.In Spreadsheet View, do one of the following:

- Click the Period/Frequency button  on the toolbar to open the dialog box, and then select **FREQUENCY**.
- Open the Timing Preferences sheet and double-click **FREQUENCY** in the Preference Name column. When the dialog box opens, the FREQUENCY preference type will already be selected.

2. Under Second Type, do one of the following:
   - Select **None** if you want to specify a clock frequency that applies to all sequential input pins that do not have a specific clock period/ frequency.
   - Select **Clock Net** if you want to specify a clock frequency for all sequential output to sequential input pins clocked by a specified net.

- Select **Clock Port** if you want to assign a clock frequency to the specified top-level clock signal port.

3. If you have chosen **Clock Net** or **Clock Port** in step 2, select one or more objects in the Available Nets/Clock Ports list.

   For nets, you can use the Available Clock list Filter and wildcard notation to create a single FREQUENCY preference for multiple clock nets. Using wildcard notation, type a string (case-sensitive) to limit the list to the clock nets you want to include, and then click Go.

   See "Using Filters and Wildcards" on page 424.

   For ports, you can use the filter to limit the list of ports displayed. You must then select a clock port from the list.

4. Type a value in MHz for frequency, and a value in nanoseconds for Hold Margin.

5. In the PAR_ADJ box, type the desired value. The PAR_ADJ option is available when **Clock Net** or **Clock Port** is selected under Second Type.

   Note

   PAR_ADJ allows you to loosen requirements for place-and-route results while tightening up the requirements reported by the TRACE static timing analysis tool. The variance in required timing values on PAR and TRACE enables you to experiment more efficiently with the strategy of purposely over-constraining your design to determine your best constraint settings. For example, if FREQUENCY is set to 250MHz and you set PAR_ADJ to 20, the software will use the simple addition formula of 250 + 20 = 270MHz to direct PAR while running TRACE on 250MHz.

6. In the CLOCK_JITTER (p-p) box, which is available for a clock port only, type a value in nanoseconds for peak-to-peak jitter on the incoming clock.

   Trace will use half of this peak-to-peak jitter value for all analysis.

7. Click **OK**.

   The Timing Preferences sheet is activated and displays the new preference in the FREQUENCY list.

Setting INPUT_SETUP and CLOCK_TO_OUT Preferences

INPUT_SETUP specifies a setup time requirement for input ports relative to a clock net. CLOCK_TO_OUT specifies a maximum allowable output delay relative to a clock. These preferences are displayed in the Timing Preferences sheet of Spreadsheet View.

To set INPUT_SETUP preferences: 1.In Spreadsheet View, do one of the following:

- Click the Input_setup/Clock_to_out button  on the toolbar to open the dialog box, and then select **INPUT_SETUP**.

- Open the Timing Preferences sheet and double-click **INPUT_SETUP** in the Preference Name column. When the dialog box opens, the INPUT_SETUP preference type will already be selected.

2. Under Second Type, select one of the following:
   - **All Ports** to specify the time delay from all input ports to the specified clock port/net.
   - **Individual Ports** to specify the time delay from one or more specified input ports to the specified clock port/net.
   - **Group** to specify the time delay from one or more specified input groups to the specified clock port/net.
3. If you have chosen Individual Ports or Group in step 2, select one or more ports in the Available Input Ports list or one or more groups in the Available Input Groups list.

   For ports, you can use wildcard notation in the Available Input Ports filter to create a single INPUT_SETUP preference for multiple ports. Type a string (case-sensitive) to limit the list to the input ports you want to include, and then click **Go**.

   See "Using Filters and Wildcards" on page 424.

   For groups, you can use the filter to limit the list of groups displayed. You must then select one or more groups from the list.
4. Under Available Clock Ports/Nets, select the destination port/net.

   You can use the Clock Ports/Nets filter to limit the list. You must then select a clock port or net from the list.
5. In the Time box, enter a delay value in nanoseconds.
6. If you also want to specify hold, enter a value in the Hold Time box.
7. To specify a multiple of the clock period for adjusting the timing analysis, type a value in the Clock Offset text box.
8. Optionally select the following:
   - **Input Delay** to specify Time as an INPUT_DELAY value
   - **PLL Phase Back** to reverse the offset direction of the phase delay programmed into a PLL
   - **SS** to indicate that the data input and clock input are source-synchronous. This will allow input jitter specified on the clock to be ignored for setup and hold timing analysis.
9. Click **OK**.

   The Timing Preferences sheet displays the new preference in the INPUT_SETUP list.

To set CLOCK_TO_OUT preferences: 1.In Spreadsheet View, do one of the following:

- Click the Input_setup/Clock_to_out button  on the toolbar to open the dialog box, and then select **CLOCK_TO_OUT**.

- Open the Timing Preferences sheet and double-click **CLOCK_TO_OUT** in the Preference Name column. When the dialog box opens, the CLOCK_TO_OUT preference type will already be selected.

2. Under Second Type, select one of the following:
   - **All Ports** to specify the time delay from all output ports to the specified clock port/net.
   - **Individual Ports** to specify the time delay from the specified output ports to the specified clock port/net.
   - **Group** to specify the time delay from the specified output group to the specified clock port/net.
3. If you have chosen **Individual Port** or **Group** in step 2, select one or more ports in the Available Output Ports list or one or more groups in the Available Output Groups list.

   For ports, you can use wildcard notation in the Available Output Ports filter to create a single CLOCK_TO_OUT preference for multiple ports. Type a string (case-sensitive) to limit the list to the output ports you want to include, and then click **Go**.

   See "Using Filters and Wildcards" on page 424.

   For groups, you can use the filter to limit the list of groups displayed. You must then select one or more groups from the filtered list.
4. Under Clock Ports/Nets, select the destination port/net.

   You can use the Clock Ports/Nets filter to limit the list of clocks displayed. You must then select a clock port or net from the list.
5. If you want to specify the output clock port with respect to which output delay needs to be measured, make your selection under Clock Out Ports.

   You can use the Clock Out Ports filter to limit the list of ports displayed. You must then select a port from the list.
6. In the Time box, enter a time delay value in nanoseconds.
7. If you also want to specify hold time, enter a value in the Min Time box.
8. To specify a multiple of the clock period for adjusting the timing analysis, type a value in the Clock Offset text box.
9. To specify a register as the starting point for the maximum delay, click the **From Cell** button. In the From Cell dialog box, you can use the filter to narrow the list. You must then select a cell from the list and click **OK**.
10. Optionally, select the following:
    - **Output Delay** to specify this as an OUTPUT_DELAY value
    - **PLL Phase Back** to reverse the offset direction of the phase delay programmed into a PLL
11. Click **OK**.

    The Timing Preferences sheet displays the new preference in the CLOCK_TO_OUT list.

Setting MULTICYCLE Preferences

MULTICYCLE allows for relaxation of previously defined PERIOD/ FREQUENCY preferences on a path. MULTICYCLE is considered in conjunction with the FREQUENCY and PERIOD preferences specified and will relax ONLY those paths that are constrained by FREQUENCY and PERIOD. These preferences are displayed in the Timing Preferences sheet of Spreadsheet View.

To set MULTICYCLE preferences:

1. In Spreadsheet View, click the Multicycle Preference button  on the toolbar.

   Alternatively, open the Timing Preferences sheet and double-click **MULTICYCLE** in the Preference Name column.

2. In the dialog box, under Type, do one of the following:

   - Select **General** if you want to relax the PERIOD or FREQUENCY preferences on a specified general path.
   - Select **Clock Net to Clock Net** if you want to relax the PERIOD or FREQUENCY preferences on a path starting from one clock net to another.
   - Select **Slow/Fast Path Exception** if you want to relax the PERIOD or FREQUENCY preferences on a path starting from one user group to another.

3. In the “From” area, do one or both of the following:

   - Click **<path_elem>** to specify the start point component for the MULTICYCLE path.
   - Click **CLKNET <Snet>** to specify the source clock net/port for the MULTICYCLE path.

   For ports and cells, you can use wildcard notation in the dialog box filters to create a single MULTICYCLE preference for multiple elements. Type a string (case-sensitive) to limit the list to the ports or cells you want to include, and then click **Go**.

   See “Using Filters and Wildcards” on page 424.

   For other elements, you can use the filter to limit the list of elements displayed. You must then select an element from the list.

4. In the “To” area, do one or both of the following:

   - Click **<path_elem>** to specify the end point component for the MULTICYCLE path.
   - Click **CLKNET <Dnet>** to specify the destination clock net/port for the MULTICYCLE path.

   If desired, use the dialog box filters, as explained in Step 3.

5. Do one of the following:

   - Select **Time** and type a delay value in nanoseconds.

- Select **X** and type a value to specify a multiple of the clock period for the clock driving the start and end point components. Select X, X_Source, or X_Dest from the drop-down list.

6. Optionally, choose a path type from the drop-down list:
   - Select READPATHS to define the multicycle paths at memory-block read-side.
   - Select SAMECLKEN to define the multicycle paths at the source/ destination register controlled by the same clock-enable signal.
   - Select CLKEN_NET to define the multicycle paths at the source/ destination register controlled by a given <net_name>.

     If you choose CLKEN_NET, specify the net name in the ClkEnNet text box.
7. Click **Add**.

   The Timing Preferences sheet displays the new preference in the MULTICYCLE list.

See Also "Modifying Timing Preferences" on page 505

- "Using Wildcard Expressions in Preferences" on page 533.

Setting MAXDELAY Preferences

MAXDELAY identifies a maximum total delay for a net, bus, or path in the design. Maximum delay preferences are displayed in the Timing Preferences sheet of Spreadsheet View.

To set a MAXDELAY preference:

1. In Spreadsheet View, click the MaxDelay Preference button  on the toolbar.

   Alternatively, open the Timing Preferences sheet and double-click **MAXDELAY** in the Preference Name column; or right-click MAXDELAY and choose **New MAXDELAY FROM TO** from the pop-up menu.
2. In the "From" area, click **<path_elem>** to specify the start point component for the MAXDELAY path.

   For ports and cells, you can use wildcard notation in the dialog box filters to create a single MAXDELAY preference for multiple elements. Type a string (case-sensitive) to limit the list to the ports or cells you want to include, and then click **Go**.

   See "Using Filters and Wildcards" on page 424.

   For other elements, you can use the filter to limit the list of elements displayed. You must then select an element from the list.
3. In the "To" area, click **<path_elem>** to specify the end point component for the MAXDELAY path.

If desired, use the dialog box filter, as explained in Step 2.

Note

It is recommended that you always specify both the start point and end point of the path. If you do not select a component for both FROM and TO, the default for the unspecified path element will include all I/O ports and registers, which is similar to the preference FREQUENCY. If the BLOCK ASYNCPATH preference is set to ON, all input-pad-to-register paths will be removed for the unspecified start point or end point, in the same way as the preference FREQUENCY.

4. In the Time text box, type a delay value in nanoseconds.
5. Click **Add**.

   The Timing Preferences sheet is activated and displays the new preference in the MAXDELAY list.

See Also ▶"Modifying Timing Preferences" on page 505

- "Using Wildcard Expressions in Preferences" on page 533
- "Setting a MAXDELAY Preference for a Net" on page 502

Setting a MAXDELAY Preference for a Net

The MAXDELAY_NET dialog box enables you to set a MAXDELAY preference for a specific net instead of a path. When you use Net, the maximum delay constraint applies to all driver-to-load connections on the net.

To set a MAXDELAY preference for a specific net:

1. In Spreadsheet View, select the Timing Preferences sheet.
2. Right-click **MAXDELAY** and choose **New MAXDELAY NET** from the pop-up menu..
3. In the dialog box, type the full name of the desired net in the name text box.
4. To include only data path delay information, choose **ON** from the dataPath Only drop-down menu.
5. In the delay text box, type a delay value in nanoseconds.
6. Click **Add**.

   The new MAXDELAY_NET preference is added to the Timing Preferences sheet under MAXDELAY.

See Also ▶"Modifying Timing Preferences" on page 505

Setting a CLKSKEWDIFF Preference

The CLKSKEWDIFF preference specifies an input skew between two top-level input clock ports.

To set a CLKSKEWDIFF preference:

1. In Spreadsheet View, select the Timing Preferences sheet.
2. Double-click **CLKSKEWDIFF** to open the dialog box.
3. Select a clock from the Available Clock Ports 1 list and one from the Available Clock Ports 2 list.

   You can use the Filter box and wildcards to narrow each of these list. Type a string (case-sensitive), and then click **Go**.
4. In the Value text box, type a value for the maximum skew between the selected top-level ports.
5. Optionally, type a value for the minimum clock skew in the Min Value text box.
6. Click **OK**.

   The new CLKSKEWDIFF preference is added to the Timing Preferences sheet.

See Also ▶"Modifying Timing Preferences" on page 505

▶ "Using Filters and Wildcards" on page 424

Setting Timing Preferences in Netlist View

Netlist View enables you to set timing preferences for selected ports and nets. The preferences are added to the Timing Preferences sheet of Spreadsheet View.

To add timing preferences for selected ports:

1. In Netlist View, click the Ports button  to open the ports tree.
2. Select the desired port, right-click, and choose a timing preference from the pop-up menu.
3. Follow the instructions for setting PERIOD and FREQUENCY, INPUT_SETUP and CLOCK_TO_OUT, MULTILCYCLE, or MAXDELAY preferences.

To add a PERIOD or FREQUENCY preference for a selected clock net:

1. In Netlist View, click the Nets button  to open the nets tree, and then expand the Clock folder.
2. Select the desired clock net, right-click, and choose **Set Clock Preference**.
3. Follow the instructions for setting PERIOD and FREQUENCY preferences.

To add BLOCK preferences for selected nets:

1. In Netlist View, click the Nets button  to open the nets tree.
2. Select the desired net, right-click, and choose **Set BLOCK Preference**.

3. Follow the instructions for setting BLOCK preferences.

The timing preferences are added to the Timing Preferences sheet of Spreadsheet View.

Setting PLL and DLL Attributes

If your design contains PLL or DLL modules, Spreadsheet View will include them on the Timing Preferences sheet, which enables you to set both PLL and DLL attributes.

To set PLL or DLL Attributes:

1. In Spreadsheet View, click the Timing Preferences tab.

   The Timing Preferences sheet shows the type, name, and attributes of your PLL/DLL module. For LatticeECP/EC, LatticeECP2, and LatticeXP designs, you can only edit the FDEL attribute. LatticeSC/M designs have more PLL and DLL attributes available for editing.

2. Double-click the Preference Value cell for the PLL or DLL preference you want to edit, and then type a new value.

The attribute values you specify in Spreadsheet View take precedence over those defined during module generation. You can also edit PLL/DLL attributes directly in the ASCII logical preference file (.lpf) using a text editor.

See "How to Design with sysCLOCK PLLs and DLLs" on page 1168 for more information, including application notes that are available from the Lattice Semiconductor web site.

See Also

- "FDEL" on page 1301
- "CLKI_FDEL" on page 1288
- "CLKFB_FDEL" on page 1287
- "CLKOS_FDEL" on page 1291
- "PHASEADJ" on page 1320
- "SPREAD_DRIFT" on page 1326
- "GSR" on page 1303
- "CLKI_DIV" on page 1288
- "CLKOS_FPHASE" on page 1291
- "CLKOS_DIV" on page 1291
- "CLKOP_DUTY" on page 1290
- "CLKOS_PHASE" on page 1291
- "ALU_LOCK_CNT" on page 1282
- "ALU_UNLOCK_CNT" on page 1283
- "SMI_OFFSET" on page 1326

▶ "LOCK_DELAY" on page 1313

Modifying Timing Preferences

Use the Timing Preferences sheet to modify or delete a timing preference.

To modify a timing preference:

1. Open the Timing Preferences sheet and double-click the preference that you want to modify. Alternatively, right-click the preference and choose **Modify** from the pop-up menu.
2. In the dialog box, modify the preference as desired and click **Update**.

Note

You can also modify values for timing preferences directly on the Timing Preferences sheet. Double-click or right-click the Preference Value cell and type a new value.

To delete a timing preference:

- On the Timing Preferences sheet, select the preference that you want to remove and press the Delete key. Alternatively, right-click the preference and choose **Delete** from the pop-up menu.

Using Timing Preference Files (.tpf)

A timing preference file (.tpf) is an editable version of your timing preferences that you can modify for the purpose of timing analysis. It is available from Timing Analysis View after placement and routing. Timing preference files enable you to experiment with different sets of timing preferences and obtain quick analysis without having to rerun Map, Place & Route.

Timing Analysis View provides a "Spreadsheet View – TPF" that displays global, timing, and group preferences. When you first open the TPF Spreadsheet View, it displays any timing preferences from the HDL source files and the active logical preference file (.lpf). You can modify these preferences in the TPF Spreadsheet View or create new ones. When you save the changes, Diamond writes them to a .tpf file that you name, and it adds the .tpf file to the Analysis Files folder.

When you are ready, you can transfer timing preferences to Spreadsheet View from the TPF Spreadsheet. You can then add them to the logical preference file with the Save command.

See "Using Timing Analysis View" on page 732 for complete information.

See Also ▶"Analyzing Static Timing" on page 694

Grouping Logical Components

Diamond enables you to establish groups of ports, cells, or ASIC blocks and create universal groups (UGROUPs) of logical elements to guide placement and routing.

Defining groups and regions should be done in the pre-mapped stage of the design flow, whenever possible. The topics that follow explain how to define logical groups using Diamond's preference-editing views.

Defining Port, Cell, and ASIC Groups

Cell, port, and ASIC groups enable you to place identical timing constraints on grouped elements. For port groups that use certain SSTL or HSTL I/O types, the defined group allows you to associate a voltage reference (Vref) for all the signals in the group.

The Group Assignment dialog box allows you to define each type of group and create multiple groups in one session. It is available from the Spreadsheet View toolbar, the Group preference sheet, and Spreadsheet View's Edit > Preference menu. It is also available from Netlist View for selected ports or registers. The groups are written to the logical preference file as DEFINE GROUP preferences.

To define a port group:

1. In Spreadsheet View, click the **Grouping Assignment button** on the toolbar.

   Alternatively, open the Group preference sheet and double-click **Define Port Group**.

2. In Group Name box at the top, type a name for the new group.
3. Under Type, select **Port**.

   All the ports in your design are displayed in the Available Ports list.

   To narrow the list of ports, select a port type from the **By** drop-down list under Group Name. The Available Ports list will be updated to include only the selected type of ports.

4. In the Available Ports list, do one of the following:
   - Select one or more ports, and then click **>** to add the instances to the Selected Ports list.
   - Use the Filter edit box to narrow the list, and click **Go**. Click click **>>** to add the entire filtered list to the Selected Ports list or select from one or more ports from the list and click **>**.
   - Use wildcard notation in the Filter edit box to create a single-line DEFINE GROUP preference that includes the wildcard expression. Type a string (case-sensitive) to limit the list, and then click **=>** to add the filtered list as a wildcard expression to the Selected Ports list.
5. To customize the color for the port group, click the **Color** button in the top right and select a color from the Select Color dialog box.

The color for the port group will persist when you run Place & Route, enabling you to identify each group within Package View when you use the View > Show Port Groups command.

6. Click **Add**.

   The Port Group appears at the bottom of the Port Assignments sheet, enabling you to set attributes for the entire group at once.

   The Group preference sheet also displays the new preference in the Define Port Group list.

To define a cell instance group: 1.In Spreadsheet View, click the Grouping Assignment button  on the toolbar.

Alternatively, open the Group preference sheet and double-click **Define Cell Group**.

2. In the Group Name box at the top, type a name for the new group.
3. Under Type, select **Cell**. All the cell instances in your design are displayed in the Available Instances list.

   To narrow the list, select **By Clock**, **By Clock Enable**, **By Sync Reset**, or **By Async Reset** in the "By" drop-down list under the Group Name, and then select one item from the list that opens on the left. The Available Instances list in the middle will be updated to include only the instances connected with the selected clock, clock enable, sync reset, or async reset.

4. In the Available Instances list, do one of the following:

   - Select one or more instances, and then click **>** to add the instances to the Selected Instances list.
   - Use the Filter edit box to narrow the list, and click **Go**. Click click **>>** to add the entire filtered list to the Selected Instances or select from one or more instances from the list and click **>**.
   - Use wildcard notation in the Filter edit box to create a single-line DEFINE GROUP preference that includes the wildcard expression. Type a string (case-sensitive) to limit the list, and then click **=>** to add the filtered list as a wildcard expression to the Selected Instances list.

5. Click **Add**.

   The Group preference sheet displays the new preference in the Define Cell Group list.

To define an ASIC block group: 1.In Spreadsheet View, click the Grouping Assignment button  on the toolbar.

Alternatively, open the Group preference sheet and double-click **Define ASIC Group**.

2. In the Group Name box at the top, type a name for the new group.
3. Under Type, select **ASIC**. All the ASIC instances in your design are displayed in the Available ASIC Instances list.

4. In the Available ASIC Instances list, select one instance. You can use the filter text box to narrow the list before making your selection. All pins for the selected instance are displayed in the Available Pins list.
5. In the Available Pins list, do one of the following:
   - Select one or more ASIC pins, and then click **>** to add them to the Selected ASIC-Pin Pairs list.
   - Use the Filter edit box to narrow the list, and click **Go**. Click click **>>** to add the entire filtered list to the Selected ASIC-Pin Pairs list or select from one or more pins from the list and click **>**.
   - Use wildcard notation in the Filter edit box to create a single-line DEFINE GROUP preference that includes the wildcard expression. Type a string (case-sensitive) to limit the list, and then click **=>** to add the filtered list as a wildcard expression to the Selected ASIC-Pin Pairs list.
6. Click **Add**.

   The Group preference sheet displays the new preference in the Define ASIC Group list.

To define a group from selected ports or registers 1.In Netlist View, do one of the following:

- Click the Ports button on the toolbar to define a port group.
- Click the Instances button on the toolbar to define a cell group.

2. Select the ports or registers that you want included in the group.
3. Right-click the selected ports or registers and choose **Define GROUP Name** from the pop-up menu.

   The Type is automatically selected in the Group Assignment dialog box. The ports or registers you selected are displayed in the Selected Ports or Selected Instances list.
4. Type a name for the group in the Group Name box.
5. Use the **>** and **<** buttons to add or delete items from the Selected Ports or Selected Instances list.
6. Click **Add**.

   The Group preference sheet displays the new preference in the Define Port Group or Define Cell Group list.

Setting Preferences for Port Groups

After you have defined port groups for your design, you can set preferences for them in Spreadsheet View.

The Map Design process will turn each port group into a PGROUP. For example, if your LPF file has the following:

Your PRF file will have:

```
DEFINE PORT GROUP "grp1" "d1" "d2";
IOBUF GROUP "grp1" IO_TYPE=LVCMOS33 BANK=3;


PGROUP "grp1" COMP "d1" COMP "d2";
LOCATE PGROUP "grp1" BANK 3;
```

To set preferences for port groups:

1. In Spreadsheet View select the Port Assignments sheet and scroll down to the bottom.

   Port groups are displayed at the bottom of the Port Assignments sheet in the Other > Groups folder.

2. For each preference that you want to assign, double-click the cell in the appropriate column and select the value from the drop-down menu.

See Also ▶"Setting Vref Locations" on page 473

- "IO_TYPE" on page 1310
- "LOCATE" on page 1227

Creating UGROUPs

The Universal Group (UGROUP) preference consists of logical components that are to be packed close together. UGROUPs can include PFU, PFF, EBR, and DSP blocks. You can create UGROUPs in Spreadsheet View, Netlist View, NCD View, and Floorplan View. UGROUPs are listed on the Group preference sheet of Spreadsheet View. After placement and routing, Floorplan View displays the group member elements with the UGROUP color fill.

Note

A UGROUP must be anchored if it includes a DSP block. If such a UGROUP is not anchored, Diamond will issue a warning and ignore the DSP block.

Creating UGROUPs in Spreadsheet View Spreadsheet View's Group preference sheet provides immediate access to the Create New UGROUP dialog box, which allows you to select components for a UGROUP and define the anchor location and bounding box.

To create a UGROUP in Spreadsheet View:

1. Select the **Group** preference sheet and double-click **UGROUP**.
2. In the Create New UGROUP dialog box, specify a name for the group and select a color.
3. To anchor the group, select **Anchor**, and then select either **Region** or **Site**.
   - If you have selected Region, select a REGION from the drop-down list. Regions must already be defined in order to appear on the list.

When Region is selected, the group will float inside the selected REGION.

- If you have selected Site, specify the column and row for the anchor. The allowable range is shown in parentheses above each box.

  When Site is selected, the specified site will serve as the northwest corner anchor location.

4. To define the size of the group's bounding box, select **BBox**, and then specify the number of columns and rows that the UGROUP's bounding box should cover. The allowable range for height and width is shown in parentheses above each box.

   The group's resource coverage is now displayed at the top under UGROUP Name: the number of LUTs, registers, and EBRs.

5. From the list of Available Instances, select those that you want included in the UGROUP.

   - Use a string and wildcards (asterisk or question mark) in the Filter text box to narrow the list.
   - Use the Ctrl key or the Shift key to select multiple instances from the full or filtered list.

6. Click the **>** button to move the selected instances to the Selected Instances list. Click the **>>** button to move all instances to the Selected instances list.

   You can further modify the group membership by moving elements between the Available Instances and Selected Instances lists.

7. Click **Add**.

   The new UGROUP is added to the Group preference sheet.

   Make sure that the UGROUP's bounding box does not overlap the bounding box of another UGROUP or REGION. Compare the dimensions on the Group sheet and the Misc sheet and modify the UGROUP if necessary.

Creating UGROUPs in Netlist View Netlist View enables you to create a UGROUP from selected instances in the netlist.

To create a UGROUP in Netlist View:

1. In Netlist View, select the Instance button  on the toolbar, and then expand the Instances folder.
2. Select the instances that you want to place in a UGROUP.
   - Use a string and wildcards in the Filter box at the top to narrow the list of instances.
   - Use the Ctrl key or the Shift key to select multiple instances from the full or filtered list.
3. Right-click the selected instances and select **Set UGROUP Preference**.

   The Create New UGROUP dialog box opens. The instances you selected are displayed in the Selected Instances list on the lower right.

4. In the boxes at the top of the dialog box, specify a name for the UGROUP and select a color.
5. To anchor the group, select **Anchor**, and then select either **Region** or **Site**.
   - If you have selected Region, select a REGION from the drop-down list. Regions must already be defined in order to appear on the list.

     When Region is selected, the group will float inside the selected REGION.
   - If you have selected Site, specify the column and row for the anchor. The allowable range is shown in parentheses above each box.

     When Site is selected, the specified site will serve as the northwest corner anchor location.
6. To define the size of the group's bounding box, select **BBox**, and then specify the number of columns and rows that the UGROUP's bounding box should cover. The allowable range for height and width is shown in parentheses above each box.

   The group's resource coverage is now displayed at the top under UGROUP Name: the number of LUTs, registers, and EBRs.
7. If desired, modify the group's membership by using the **<** and **>** buttons to move elements between the Available Instances and Selected Instances lists.
8. Click **Add**.

   The new UGROUP is added to the Group preference sheet.

   Make sure that the UGROUP's bounding box does not overlap the bounding box of another UGROUP or REGION. Compare the dimensions on the Group sheet and the Misc sheet and modify the UGROUP if necessary.

Creating UGROUPs in Floorplan View In Floorplan View, you can draw a bounding box for a new UGROUP, and then select the instances to be included. For routed designs, you can create a UGROUP from selected components displayed on the layout.

To create a UGROUP bounding box and add instances:

1. Choose **Tools > Floorplan View**.
2. Zoom in, as needed, to the desired area on the floorplan layout.
3. Right-click and choose **Create UGROUP BBox**.

   The mouse pointer changes from an arrow to a cross symbol.

4. Draw a rectangle around the sites where you want to place the UGROUP. Make sure that the bounding box does not overlap the bounding box of another UGROUP or REGION.

   Note

   For certain MachXO2 devices, the first EBR site on the left is positioned at column 1 and cannot be included in an anchored group or region that contains SLICEs. The column 1 EBR site is not aligned with SLICEs, which begin at column 2. Affected devices include LCMXO2-1200/4000/7000/10000.

   When you release the mouse button, the Create New UGROUP dialog box opens. In the Anchor and BBox sections, it shows the anchor site location and the bounding box size, based on the rectangle you drew on the floorplan. At the top, under UGROUP Name, it shows the resources covered by the bounding box: LUTs, registers, and EBRs.

   Note

   A UGROUP must be anchored in order to be displayed in Floorplan View.

5. In the boxes at the top of the dialog box, specify a name for the UGROUP and select a color.
6. From the list of Available Instances, select those that you want included in the UGROUP.
   - Use a string and wildcards (asterisk or question mark) in the Filter text box to narrow the list.
   - Use the Ctrl key or the Shift key to select multiple instances from the full or filtered list.
7. Click the **>** button to move the selected instances to the Selected Instances list. Click the **>>** button to add all instances to the Selected Instances list.

   You can further modify the group membership by moving elements between the Available Instances and Selected Instances lists.
8. Click **Add**.

   The new UGROUP is displayed in Floorplan View with a bounding box in the color you selected. The new UGROUP is added to the Group preference sheet in Spreadsheet View.

To create a UGROUP from selected placed components in Floorplan View:

1. In Floorplan View, zoom in to the desired area as needed.
2. Select the placed components that you want to include in the UGROUP. These can include PFU or PFF slices, EBR blocks, or DSP blocks.

   Note

   To view the precise logical identifiers for a PFU or PFF slice, select it on the layout and cross-probe to Netlist View.

3. Right-click the selected components and choose **Create UGROUP**.

   The Create New UGROUP dialog box opens. In the Selected Instances list, it shows the placed components from the blocks you selected in Floorplan View.

4. In the boxes at the top of the dialog box, specify a name for the UGROUP and select a color.

5. To select a different anchor location, select either **Region** or **Site**.

   - If you have selected Region, select a REGION from the drop-down list. Regions must already be defined in order for them to appear on the list.

     When Region is selected, the group will float inside the selected REGION.

   - If you have selected Site, specify the column and row for the anchor. The allowable range is shown in parentheses above each box.

     When Site is selected, the specified site will serve as the northwest corner anchor location.

6. To define a different size for the group's bounding box, specify the number of columns and rows that the UGROUP's bounding box should cover. The allowable range for height and width is shown in parentheses above each box. Make sure that the bounding box does not overlap the bounding box of another UGROUP or REGION.

   The group's revised resource coverage is now displayed at the top under UGROUP Name: the number of LUTs, registers, and EBRs.

7. Click **Add**.

   The new UGROUP is displayed in Floorplan View with a bounding box in the color you selected, and it is added to the Group preference sheet in Spreadsheet View.

Creating UGROUPs in NCD View After placement and routing, you can create a UGROUP from selected instances in NCD View.

To create a new UGROUP in NCD View:

1. Select NCD View, and then expand the Instances folder.
2. Select the instances that you want to place in a UGROUP:

   - Use a string and wildcards in the Filter box at the top to narrow the list of instances.
   - Use the Ctrl key or the Shift key to select more than one instance from the full or filtered list.

   Note

   To view the precise logical identifiers for a PFU or PFF slice, select it and cross-probe to Netlist View.

3. Right-click the selected instances and select **Set UGROUP Preference**.

The Create New UGROUP dialog box opens. In the Selected Instances frame, it lists the logical components of the instances you selected.

4. In the boxes at the top of the dialog box, specify a name for the UGROUP and select a color.
5. To anchor the group, select **Anchor**, and then select either **Region** or **Site**:
   - If you have selected Region, select a REGION from the drop-down list. Regions must already be defined in order to appear on the list.

     When Region is selected, the group will float inside the selected REGION.
   - If you have selected Site, specify the row and column for the anchor. The allowable range is shown in parentheses above each box.

     When Site is selected, the specified site will serve as the northwest corner anchor location.
6. To define the size of the group's bounding box, select **BBox**, and then specify the number of columns and rows that the UGROUP's bounding box should cover. The allowable range for height and width is shown in parentheses above each box.

   The group's resource coverage is now displayed at the top under UGROUP Name: the number of LUTs, registers, and EBRs.
7. If desired, modify the group's membership by using the **<** and **>** buttons to move elements between the Available Instances and Selected Instances lists.
8. Click **Add**.

   The new UGROUP is added to the Group preference sheet.

   Make sure that the UGROUP's bounding box does not overlap the bounding box of another UGROUP or REGION. Compare the dimensions on the Group sheet and the Misc sheet and modify the UGROUP if necessary.

Grouping Components Along a Net

Both Netlist View and NCD View enable you to create a UGROUP from selected nets. The selected nets must be connected to NGD components that are appropriate for UGROUPs. These types of components include PFU, PFF, EBR, and DSP blocks. A selected net in NCD View, after placement and routing, can be used to identify a set of NCD components into which UGROUP-appropriate NGD components are packed.

To group components along a net in Netlist View or NCD View:

1. Open Netlist View or NCD View, and then select the Nets view.
2. Select one or more nets, right-click, and choose **Set UGROUP Preference**.

In the Create UGROUP dialog box, the Selected Instances list displays any appropriate components that are connected to the selected nets.

Note

If no appropriate instances for a UGROUP are connected to the selected nets, the Selected Instances list will be empty.

3. In the boxes at the top of the dialog box, specify a name for the UGROUP and select a color.
4. To anchor the group, select **Anchor**, and then select either **Region** or **Site**:
   - If you have selected Region, select a REGION from the drop-down list. Regions must already be defined in order to appear on the list.

     When Region is selected, the group will float inside the selected REGION.
   - If you have selected Site, specify the row and column for the anchor. The allowable range is shown in parentheses above each box.

     When Site is selected, the specified site will serve as the northwest corner anchor location.
5. To define the size of the group's bounding box, select **BBox**, and then specify the number of columns and rows that the UGROUP's bounding box should cover. The allowable range for height and width is shown in parentheses above each box.

   The group's resource coverage is now displayed at the top under UGROUP Name: the number of LUTs, registers, and EBRs.
6. If desired, modify the group's membership by using the **<** and **>** buttons to move elements between the Available Instances and Selected Instances lists.
7. Click **Add**.

   The new UGROUP is added to the Group preference sheet.

See Also ▶"Grouping Components Along an Individual Signal" on page 515

Grouping Components Along an Individual Signal

In Floorplan View, you can group components along an individual signal of a timing path. You can do this by cross-probing a signal from Timing Analysis View to Floorplan View. From Floorplan View, you can then select the components along the displayed connection to create a new UGROUP.

To group components along an individual signal:

1. After placement and routing, choose **Tools > Timing Analysis View**.
2. In the timing preference list in the lower left pane, expand Analysis Results, and then select the desired timing preference.

   The path table opens in the upper right pane.

3. Select a signal from the path table, right-click, and choose **Show in > Floorplan View**.

   The connection is highlighted in Floorplan View.

4. While holding the **Ctrl** key, click each component related to the net.
5. Right-click the selected components and choose **Create > UGROUP**.

   The Create New UGROUP dialog box displays the anchor location and bounding box size of the components you selected, and it lists the instances in the Selected Instances pane.

6. In the top part of the dialog box, assign a name to the UGROUP. Click the color box to assign a different color.
7. Click **Add**.

   The new UGROUP is displayed in Floorplan View with a bounding box in the color you selected. The new UGROUP is added to the Group preference sheet in Spreadsheet View.

See Also ▶"Using Timing Analysis View" on page 732

Modifying UGROUPs

Use the Edit UGROUP Property dialog box to quickly modify the name, color, and membership of a group preference. You can access the dialog box from the Group preference sheet and from Floorplan View. You can also use Floorplan View to delete a UGROUP, or you can use the Group preference sheet to delete a UGROUP or UGROUP elements.

To modify a group preference:

1. To access the Edit UGROUP Property dialog box, do one of the following:
   - In the Group sheet of Spreadsheet View, double-click the group you want to modify or right-click the group and choose **Modify**.
   - In Floorplan View, zoom in to the UGROUP and click the border or an empty space inside so that the group's border is highlighted. Right-click and choose **Edit UGROUP**.
2. In the dialog box, make the desired changes and click **Update**.

To remove elements from a group on the Group preference sheet:

1. Select the Group preference sheet and expand the desired UGROUP.
2. Select the group elements you want to remove, right-click, and choose **Remove UGroup Elements**.

To delete a group preference on the Group sheet:

- Right-click the group you want to delete and choose **Delete**, or press the Delete key.

To delete a UGROUP using Floorplan View:

1. Zoom in to the UGROUP and click the border or an empty space inside so that the group's border is highlighted.
2. Right-click and choose **Remove UGROUP**.

Note

A UGROUP that originated in the HDL source does not get removed from the logical preference file (.lpf) when you simply reverse any modifications that were made in Diamond. To restore the UGROUP to its HDL settings, you must remove the UGROUP from the .lpf file. To remove the UGROUP completely, you must delete it from both the .lpf file and the HDL source.

See Also "Creating UGROUPs" on page 509

- "Assigning a UGROUP to a REGION" on page 520
- "Removing a REGION Assignment from a UGROUP" on page 521

Resizing or Moving a UGROUP in Floorplan View

You can use your mouse to resize a UGROUP in Floorplan View or change its anchor location.

To resize a UGROUP using Floorplan View:

1. In Floorplan View, zoom in to the UGROUP and select it by clicking the border or an empty space inside. The border should be highlighted. You should be able to see the small circles on the UGROUP's border—one in each corner and one on each side.

   Note

   By default, UGROUPs are displayed in Floorplan View. If they do not appear on the layout, click the  button on the vertical toolbar.

2. Position your mouse pointer over one of the corner or side circles.

   The cursor symbol changes to a double-headed arrow.

3. Drag the border outward or inward until the bounding box contains the sites you want to include.

   Floorplan View displays the new bounding box dimensions. The Group sheet in Spreadsheet View shows the updated dimensions.

To change the UGROUP's anchor location:

1. Zoom out until you can easily view the current location on the layout and the targeted location.
2. Click the UGROUP's border or an empty space inside it so that the border is highlighted.

   The crossed arrows symbol should immediately appear. If it does not, move your mouse pointer very slightly until it does appear.

3. Hold down the left mouse button, drag the UGROUP box to the targeted location, and then release the mouse button.

   Floorplan View displays the UGROUP in its new location. The Group sheet in Spreadsheet View shows the updated location.

See Also ▶"Creating UGROUPs" on page 509

- "Assigning a UGROUP to a REGION" on page 520

Setting Miscellaneous Preferences

Spreadsheet View's Misc Preferences sheet enables you to assign Vref locations and define REGIONs. Double-click the preference type that you want to create to open the dialog box. Or double-click a listed Vref or REGION preference to open the Edit dialog box. A list of current preferences is displayed under each preference type.

The Misc Preference sheet also lists modules and any sites that have been reserved with the PROHIBIT preference. A site reserved with the PROHIBIT preference will not be used in placement and routing. The PROHIBIT SITE preference can be set using Floorplan View, Device View, Package View, or the Pin Assignments sheet of Spreadsheet View.

See Also ▶"Reserving Resources" on page 524

- "Setting Vref Locations" on page 473
- "Defining REGIONs" on page 518

Defining REGIONs

A REGION preference sets aside a fixed area of the device for the placement of UGROUPs or for reserving resources. A REGION that is set aside for the placement of UGROUPs instructs the placer to place the groups within the area specified by the REGION's bounding box. A reserved REGION instructs the placer to avoid all resources within the REGION's bounding box.

When you create a REGION, you identify a northwest corner as the anchor location and specify the width and height of the bounding box. Since a REGION always defines a physical rectangle inside the device (DEVSIZE), its bounding box will be measured by contiguous rows and columns regardless of the types of components they contain. This means that it can include such components as PFUs, PFFs, EBRs, and DSP blocks.

Note

For certain MachXO2 devices, the first EBR site on the left is positioned at column 1 and cannot be included in an anchored group or region that contains SLICEs. The column 1 EBR site is not aligned with SLICEs, which begin at column 2. Affected devices include LCMXO2-1200/4000/7000/10000.

You can create a new REGION using Spreadsheet View's Misc Preferences sheet, or you can use your mouse to draw a REGION in Floorplan View. REGIONs are displayed with solid colored borders on the layout.

Note

Unless a REGION is reserved or contains one or more UGROUPs, it will be ignored by the Place & Route process.

For more information about REGIONs and other physical preferences, refer to the "Preferences" on page 1195 section of the Constraints Reference Guide.

See Also

- "Creating a REGION" on page 519
- "Assigning a UGROUP to a REGION" on page 520
- "Reserving REGIONs" on page 525
- "Using Incremental Design Flow" on page 651

Creating a REGION

The Create New REGION dialog box is available from the Misc Preferences sheet of Spreadsheet View and from Floorplan View. It enables you to create a REGION preference for the placement of UGROUPs or a reserved REGION that will be not be used in the next placement process. For LatticeECP3 devices, the dialog box also allows you to create clock REGIONs for assigning secondary clocks.

To create a new REGION:

1. Do one of the following to open the Create New REGION dialog box:
   - In Spreadsheet View, select the **Misc Preferences** sheet, and then double-click **REGION**. Alternatively, right-click and choose **New Region**.
   - In Floorplan View, right-click an empty area of the floorplan layout and choose **Create > REGION**. Draw a rectangle around the area of the layout that you want to include in the REGION, and then release the mouse button.
2. At the top of the Create New REGION dialog box, give the REGION a descriptive name and select a color to identify the border.
3. To reserve the REGION and prevent its resources from being used, select the **Prohibit** option.

   Note

   You can also reserve the REGION later by right-clicking it on the Floorplan View layout and choosing **Prohibit**. When you reserve a REGION, it will not be available for groups.

4. If you used Spreadsheet View, type the row and column that will identify the northwest corner of the bounding box. This information is required.

Make sure that the values you type are within the available range displayed.

If you used Floorplan View, the Anchor text boxes will already be populated.

5. If you used Spreadsheet View, type the width and height for the bounding box in the BBox text boxes. This information is also required. Make sure that the values are no greater than the maximum values displayed by default.

   If you used Floorplan View to draw a REGION, the BBox text boxes will already be populated.

   After the anchor and bounding box are defined, the dialog box displays the area's resource coverage near the top, including LUTs, registers, and EBRs.

   Note

   DEVSIZE, which is automatically selected, means that all contiguous rows and columns are counted for the bounding box regardless of the types of resources they contain. Since a REGION's bounding box is always considered DEVSIZE, this check box cannot be cleared.

6. Click **Add**.

   The Misc Preferences spreadsheet displays the new REGION. Floorplan View displays the REGION with a solid border. If the REGION is reserved, the sites will be marked with a cross-hatch pattern.

   Examine the Misc sheet and the Group sheet to make sure that the new REGION does not overlap the bounding box of any other REGION or UGROUP.

   Note

   By default, REGIONs are displayed in Floorplan View. If they do not appear on the layout, click the Display Regions button  on the Floorplan View toolbar.

See Also ▶"Assigning a UGROUP to a REGION" on page 520

Assigning a UGROUP to a REGION

After you have created a REGION, you can anchor a new or existing UGROUP to it. The UGROUP will be placed in the designated REGION during placement and routing.

To assign an existing UGROUP to a REGION:

1. Do one of the following to open the Edit UGROUP Property dialog box:
   - In Spreadsheet View, select the Group preference sheet and double-click the UGROUP you want to assign. Alternatively, right-click and choose **Modify**.
   - In Floorplan View, zoom in to the UGROUP you want to assign, right-click it, and choose **Edit UGROUP**.

2. In the dialog box, select **Anchor**, and then select **Region**.

   When you select the Region option, the row and column become unavailable. If the UGROUP was originally anchored to a site, the anchor location will be removed.

3. Select the desired REGION from the drop-down list.

4. Click **Update**.

   The REGION assignment for the UGROUP is displayed on the Group preference sheet.

   After placement and routing, when “Display Placement Groups” is turned on, Floorplan View will display the group member elements within the REGION. Zoom in to view the individual slices, EBRs, or DSP blocks that contain the group elements. These blocks will be displayed with the UGROUP color fill.

See Also ▶“Creating UGROUPs” on page 509

▶ “Displaying Assignments and Connections” on page 433

Removing a REGION Assignment from a UGROUP

Use the Edit UGROUP Property dialog box to remove a REGION assignment from a UGROUP.

To remove a REGION assignment from a UGROUP:

1. In the Group sheet of Spreadsheet View, double-click the UGROUP you want to edit. Optionally, right-click the UGROUP and choose **Modify**.

2. Do one of the following:

   - To remove the REGION assignment and make the UGROUP a floating group, clear the Anchor selection.
   - To remove the REGION assignment and anchor the UGROUP to a specific site, select **Site** instead of Region and specify the row and column for the anchor location.

3. Click **Update**.

   The Group preference sheet shows the modified attributes of the UGROUP.

Defining a Clock REGION

Clock REGIONs enable you to geographically control the distribution of control signals to secondary routing and to ensure that the logic that uses these control signals is placed in the same geographic area. Devices such as the LatticeECP3 are divided into specific clock REGIONs. You can specify these REGIONs, instead of quadrants, for secondary routing. Diamond enables you define a clock REGION from Spreadsheet View or from Floorplan View. After you have defined the clock REGION, you can select it from the Quadrant column of Spreadsheet View for a secondary clock.

To define a new clock REGION:

1. Do one of the following:
   - In the Misc Preferences sheet of Spreadsheet View, right-click **REGION** and select **New Clock Region**.
   - In Floorplan View, right-click an empty area of the layout and choose **Create > CLOCK REGION**.
2. In the Create New Clock Region dialog box, specify the REGION name, anchor location and bounding box in the text boxes provided.
3. Use the Color box to change the border color for the new clock REGION
4. Click **Add**.

   The new clock REGION is added to the preference sheet. It is also displayed in Floorplan View with the color you assigned it.

To select the clock REGION for secondary routing:

- In the Clock Resource sheet of Spreadsheet View, select **Secondary** in the Selection Column for the desired clock, and then select the newly defined clock REGION from the Quadrant column.

Note

The clock REGION <clock_region_name> <region_clock_space> preference must precede the USE SECONDARY preference that references it in the logical preference file (.lpf).

See Also ▶"Setting Clock Preferences" on page 481

Editing a REGION Preference

Use the Edit Region Property dialog box to change the name, color, anchor position or bounding box of a REGION or clock REGION. You can access the dialog box from Spreadsheet or from Floorplan View. You can also edit the bounding box directly in Floorplan View.

To edit a REGION preference:

1. Do one of the following to open the Edit Region Property dialog box or the Edit Clock Region Property dialog box:
   - In Spreadsheet View, select the Misc Preferences sheet and double-click the REGION you want to edit. Optionally, right-click the REGION and choose **Modify**.
   - In Floorplan View, select the REGION you want to edit by clicking the outer border or an empty space inside the REGION. The border should be highlighted. Right-click and choose **Edit REGION** or **Edit CLOCK REGION**.
2. Modify the properties as desired and click **Update**.

   The Misc Preferences sheet shows the REGION's modified attributes.

To edit a REGION bounding box in Floorplan View:

1. Zoom in sufficiently so that you can see the small circles on the REGION's border—one in each corner and one on each side.
2. Hold your mouse above one of the circles until the double-headed arrow symbol appears.
3. Drag the sides or corners of the REGION to resize it to the desired dimensions.

See Also ▶"Assigning a UGROUP to a REGION" on page 520

- "Removing a REGION Assignment from a UGROUP" on page 521
- "Moving a REGION in Floorplan View" on page 523

Moving a REGION in Floorplan View

You can change the physical location of a REGION in Floorplan View by dragging it to a different location on the layout.

To move a REGION:

1. In Floorplan View, use the zoom tools as needed so that both the REGION you want to move and the desired new location are clearly visible.

   Note

   By default, REGIONs are displayed in Floorplan View. If they do not appear on the layout, click the Display Regions button on the Floorplan View toolbar.

2. Click the REGION's border or an empty space inside it to select it.

   The border of the REGION becomes highlighted.
3. Hold the pointer above an empty space inside the REGION until the crossed arrows symbol appears.
4. Press the left mouse button and drag the REGION to the desired location.
5. Release the mouse button.

See Also ▶"Editing a REGION Preference" on page 522

- "Assigning a UGROUP to a REGION" on page 520
- "Removing a REGION Assignment from a UGROUP" on page 521

Removing a REGION

You can use the Misc Preferences sheet in Spreadsheet View to quickly remove a REGION from your design. Alternatively, you can remove a REGION using Floorplan View.

To remove a REGION using Spreadsheet View:

1. In Spreadsheet View, select the **Misc Preferences** sheet.
2. Right-click the REGION you want to remove, and choose **Delete**. Alternatively, select the REGION and press the Delete key.

   The REGION is removed from the preference sheet and will no longer be displayed in Floorplan View.

To remove a REGION using Floorplan View:

1. Click the border of the REGION or an empty space inside it so that the border is highlighted.

   Note

   By default, REGIONs are displayed in Floorplan View. If they do not appear on the layout, click the Display Regions button  on the Floorplan View toolbar.

2. Right-click and choose **Remove REGION**.

   The REGION is removed from the floorplan layout and from the Misc Preferences sheet of Spreadsheet View.

See Also ▶"Removing a REGION Assignment from a UGROUP" on page 521

Reserving Resources

The Prohibit command enables you to specify device resources that you do not want the placer and router to use. You can reserve specific pins or sites using Package View, Device View, or Floorplan View. You can also reserve an entire area of the chip that you do not want used. When you use the prohibit command, these resources are avoided in the next Place & Route process.

See Also ▶"PROHIBIT" on page 1242

▶ "Defining REGIONs" on page 518

Reserving Sites

You can use Spreadsheet View, Package View, Device View, or Floorplan View to specify sites that you do not want to be used in the next Place & Route process. All reserved sites are listed in the PROHIBIT SITE section of Spreadsheet View's Misc Preference sheet.

To reserve sites in Spreadsheet View:

1. Select the Pin Assignments sheet, and then select the pins that you do not want to be used in placement and routing.
2. Right-click the selected pins and choose **Prohibit Pin(s)**.

Each reserved pin is tagged with the “Prohibit” label on the Pin Assignments sheet.

To reserve sites in Package View:

1. In the device pin layout of Package View, select the pins that you do not want to be used in placement and routing.
2. Right-click the selected pins and choose **Prohibit**.

   Each reserved pin is highlighted with the Prohibit color to indicate that it will not be used.

To reserve sites in Device View:

1. Expand the folder that contains the desired resources.
2. Select the sites that you want to reserve, right-click, and choose **Prohibit**.

   Each reserved site is tagged with the “Prohibited” label.

To reserve sites in Floorplan View:

1. Zoom in to the area that contains the sites that you want to reserve. The borders should be clearly visible. For PFF or PFU blocks, zoom in so that the individual slices are visible.
2. Select the sites that you want to reserve, right-click, and choose **PROHIBIT Site**.

   Each reserved site is displayed with a gray pattern.

When you save the changes, each reserved site is added as a PROHIBIT SITE preference in the .lpf file and will be avoided in the next Place & Route process.

See Also ▶“PROHIBIT” on page 1242

▶ “Reserving REGIONs” on page 525

Reserving REGIONs

The Reserve REGION option is very useful for partitioning large designs or for setting aside areas that are not to be used in the Placement and Routing process.

There are two ways to reserve a REGION: use the Prohibit option in the New REGION or Edit REGION dialog box or use the Prohibit command for an existing REGION in Floorplan View.

Note

A prohibited REGION cannot contain UGROUPs. Before reserving the REGION, make sure that no UGROUPs have been assigned to it.

To reserve a REGION using the dialog box:

1. Do one of the following:
   - Create a new REGION in Spreadsheet View or in Floorplan View to open the New REGION dialog box.
   - Double-click an existing REGION in Spreadsheet's Misc Preferences to open the Edit Region Property dialog box. Optionally, right-click the REGION and choose **Modify**.
   - In Floorplan View, right-click the selected REGION and choose **Edit REGION** to open the Edit Region Property dialog box.
2. In the dialog box, select the **Prohibit** option.
3. Click **Add** for a new REGION or **Update** for an existing REGION.

To reserve an existing REGION in Floorplan View

- Right-click the REGION on the floorplan layout and choose **PROHIBIT REGION**.

In Floorplan View, the reserved REGION is displayed with a gray pattern overlay. In Spreadsheet View, the REGION is tagged with the Prohibit label on the Misc Preferences sheet. When you save the changes, each reserved REGION is added as a PROHIBIT REGION preference in the .lpf file and will be avoided in the next Place & Route process.

See Also ▶"PROHIBIT" on page 1242

Releasing Reserved Resources

After reserving sites or REGIONs, you can easily free them up to be used for placement and routing.

To release reserved sites in Spreadsheet View, do one of the following:

- On the Pin Assignments sheet, select the prohibited sites that you want to release. Right-click and choose **Clear** or press the Delete key.
- On the Misc Preferences sheet, select the sites from the PROHIBIT SITE list that you want to release. Right-click and choose **Delete** or press the Delete key.

To release reserved sites in Package View or Device View:

- Select the reserved sites that you want to release, right-click, and choose **Release**.

To release reserved sites in Floorplan View:

1. Zoom in so that the sites are clearly visible on the layout.
2. Select the reserved sites that you want to release, right-click, and clear the **PROHIBIT Site** command from the pop-up menu.

To release a reserved REGION using Floorplan View:

- Select the reserved REGION that you want to release, right-click and clear the **PROHIBIT REGION** command from the pop-up menu.

To release a reserved REGION using Spreadsheet View:

1. On the Misc Preferences sheet, select the Preference Value cell for the REGION that you want to release.
2. Right-click and choose **Clear Prohibit**.

Setting Preferences in a Text Editor

You can create and edit preferences manually in the logical preference file using a text editor. When setting preferences manually, adhere to the guidelines outlined in "Preference Syntax Guidelines and Conventions" on page 527.

To set preferences in a text editor:

1. In Diamond, select the File tab on the left.
2. Select the desired implementation. If the selected implementation is not the active one, right-click and choose **Set as Active Implementation**.
3. Expand the Constraints Files folder inside the implementation directory, and then do one of the following:
   - Double-click the name of the .lpf file to open it in the Diamond Source Editor or the default editor.

     The logical preference file opens in Source Editor on the right.
   - Right-click the name of the .lpf file and choose **Open With**. Select the desired external editor and click **OK**.

     The logical preference file opens in the external editor.
4. Modify the preferences or add new ones as desired, and then choose **File > Save**.

See Also "Preference Flow" on page 376

"Preferences" on page 1195

Preference Syntax Guidelines and Conventions

The following guidelines explain basic syntax rules for editing preferences directly in the logical preference file and shows the conventions for character and units usage. The specific syntax for each preference is provided in the "Preferences" on page 1195 section of the Constraints Reference Guide.

- Preferences are not case sensitive. They can be typed in upper or lower case characters.

- Comment strings begin with either # or //, and continue to the end of the line.
- All statements are terminated with a semicolon, and can span multiple lines.
- User-supplied identifiers may optionally be enclosed in double quotes.
- Identifiers that start with a number must be enclosed in double quotes.
- Any identifier that is the same as a logical preference keyword must be enclosed in double quotes.
- Wildcard patterns cannot begin with a number.
- The keyword CELL refers to hierarchical instance names from the netlist. The hierarchy delimiter is a forward slash (/).
- The keyword PORT refers to top-level I/O signal names.
- The keyword PIN refers to hierarchical instance pin names.
- The keyword NET refers to hierarchical net names.
- A time value may be entered as 0–1,000,000,000 nanoseconds (entered as **NS**), 0–1,000,000 microseconds (entered as **US**), or 0–1,000 milliseconds (entered as **MS**).
- A frequency value may be entered as 0–1,000 MHz or 0– 1,000,000 KHz.
- Names of sites, nets, buses, and other design elements are case sensitive.
- If an object name in the design is a keyword in a preference, or if the name begins with anything other than a valid character, you must use quotes around the word. Valid start characters are:

  A-Z, a-z, [,], _

  For example, if the site name is 11R7, you must use quotes around the word in the preference statement: LOCATE COMP xxx SITE “11R7”
- Names may not start with a plus '+' or a minus '-' sign. If you have a name that must begin with either a plus or minus sign character, you can still do so by enclosing the name in quotes. For example, the following preference would now result in a syntax error because +any_name begins with a plus sign.

  `LOCATE COMP +any_name SITE R3C2;` (results in syntax error)

  `LOCATE COMP "+any_name" SITE R3C2;` (acceptable)
- Multiple preferences applying to the same design element that occur in the same or multiple input .prf files are subject to the rules of preference conflict resolution. See “Preference Conflict Resolution” on page 395. This generally means that preferences that are input last take precedence, overriding previously specified like preferences completely.

Conventions The following conventions are used in the preference syntax:

- Keywords are **boldfaced**.
- ( ) parentheses are for grouping

- [ ] square brackets enclose optional constructs.
- | pipe lines indicate “or”.
- + plus sign means the preceding construct is repeated one or more times.
- < > angle brackets enclose non-terminal symbols and identifiers.
- All other characters and punctuation are literal.

Units ▶Time: ms, us, ns, ps, fs

- Temperature: C, F, K
- Voltage: v
- Capacitance: F, mF, uF, nF, pF, fF

See Also ▶“The Physical Preference File” on page 405

- “Preference Keywords” on page 529

Preference Keywords

Any object name or identifier that is the same as a preference keyword must be enclosed in double quotation marks. Otherwise, the design will fail with syntax errors when you map the design. If you use any of the following preference keywords as identifiers, make sure that you enclose the name in double quotation marks.

@DEFINE
@IFDEF
@ENDIF
@IFNDEF
@ELSE

AFTER
ALLNETS
ALLPATHS
ALLPORTS
ASCII
ASIC
ASYNCPATHS
AUTO
AUTOMOTIVE

BANK
BBOX
BEFORE
BIN
BLKNAME
BLOCK
BOTTOM
BUS

C
CELL

CHECKSUM
CLK_FACTOR
CLK_OFFSET
CLKDELAY
CLKEN_NET
CLKNET
CLKOUT
CLKPORT
CLKREG
CLKSKEWDIFF
CLKSKEWDISABLE
CLOCK_JITTER
CLOCK_TO_OUT
CLOCKLOAD_ONLY
CLOCKPATHS
CLOCKWISE
CLOSED
COL
COMMERCIAL
COMP
COUNTERCLOCKWISE
CUSTOM_IDCODE

DATA_PATH
DATAPATH_ONLY
DCS
DEFINE
DEL
DELAY
DEVSIZE
DIN
DOUT
DYNAMIC

EDGE
EDGE2EDGE
ENABLE_PATH
ENABLEPATHS
ENCRYPTION
END
ENDPOINT
EXCEPT

F
FALSE
FBEXTDELAY
FF
FREQUENCY
FROM

GROUP
GSR_NET

HALFLINE
HEX
HGROUP
HIGH
HOLD
HOLD_MARGIN
HORIZONTAL

INDUSTRIAL
INPUT
INPUT_DELAY
INPUT_SETUP
INTERCLOCKDOMAIN
IOBUF
IOS

JITTER
JTAGPATHS

K
KHZ

LEFT
LEVELMODE
LOAD
LOCATE
LOCK
LONGLINE
LOW

MACO
MACRO
MAX
MAXDELAY
MAXSKEW
MF
MHZ
MIN
MODULE
MS
MULTICYCLE

NET
NF
NOMINAL
NORETIME
NS

OPEN
ORIENTATION
OUT
OUTPUT
OUTPUT_DELAY
OUTPUTPATHS

PAR_ADJ
PATH
PATHS
PCM
PERCENT
PERIOD
PF
PFUSIZE
PGROUP
PIN
PLACEMENT
PLL_PHASE_BACK
PORT
PRIMARY
PRIMARY2EDGE
PRIORITIZE
PROCESS
PROHIBIT
PS
PURE

QUADRANT_BL
QUADRANT_BR
QUADRANT_TL
QUADRANT_TR

RAIL
RANGE
RD_DURING_WR_PATHS
READPATHS
REG
REGION
RESERVE
RESETPATHS
RETIME_EFFORT
RETIME_ROUTE_DETOUR_SEVERITY
RETIME_STRATEGY
RIGHT
RLOC
ROUTING
ROW
RVL_ALIAS

SAMECLKEN
SCHEMATIC
SECONDARY
SETUP
SIDE
SITE
SITEPIN
SPINE
SS
SSO
START

STARTPOINT
SYNCPATHS
SYSCONFIG
SYSTEM_JITTER

TEMPERATURE
THROUGH
TO
TOP
TRACEID
TRUE
TWR_REPORT_LIMIT
TYPE

UF
UGROUP
UNIQUE_ID
US
USE
USERCODE

V
VCC_DERATE
VCC1P2_DERATE
VCCAUX_DERATE
VCCIO
VCCIO_DERATE
VCC_NOMINAL
VERTICAL
VOLTAGE
VOLTAGEIO
VREF

X
X_DEST
X_SOURCE

See Also ▶"Preference Syntax Guidelines and Conventions" on page 527

Using Wildcard Expressions in Preferences

Wildcards are useful in designs where bus and net names are very similar. They can be especially beneficial for large designs, because they significantly reduce the number of preferences and conserve memory. This topic describes preference wildcard support and the usage and syntax of wildcard characters and integer ranges.

Preferences that Support Wildcards The following table shows the preferences that support wildcard expressions and the elements that are supported for each preference. For many of these preferences, the applicable

dialog boxes provide filters that allow wildcard notation. These dialog boxes are available from Spreadsheet View. For more information, see "Using Filters and Wildcards" on page 424.

Table 38:

Preference	Wildcard Support
BLOCK	Nets, buses, ports, cells
CLOCK_TO_OUT	Ports
DEFINE GROUP	Ports, cells, ASIC pins
FREQUENCY	Nets
INPUT_SETUP	Ports
IOBUF	Groups
MAXDELAY	Ports, cells
MULTICYCLE	Ports, cells
OUTPUT LOAD	Ports
PERIOD	Nets, ports
SSO	Ports
USE SECONDARY	Nets

Wildcard Characters The logical preference language allows the following set of wildcard characters to be used in the specification of cell, port, and pin names:

***** Match any (zero or more) characters.

? Match any single character.

?+ Match one or more characters.

{ . . . } Match any single character in the enclosed list or range. A list is a string of characters separated by commas. A lexical range is two characters separated by a dash (-), and includes all the ASCII characters in between. An integer range is two integers separated by a colon (:) and includes all integers between the two.

{ . . . }+ Match one or more occurrences of any character from the preceding list.

Range Expansion The logical preference language syntax allows the specification of integer ranges. This feature, used in conjunction with wildcard expansion, is a powerful tool that can simplify the task of writing a preference file.

General syntax is as follows:

```
<bus_range> := {<range_list>}
```

```
<range_list> := <range_spec> | <range_list>,<range_spec>
<range_spec> := <upper>:<lower> | <bit_position>
<upper> := integer
<lower> := integer
<bit_position> := integer
```

Examples of Wildcard and Range Expansion The following examples demonstrate some of the most common uses of wildcards. These examples refer to logical preferences.

Wildcards for Timing-Related I/Os

```
INPUT_SETUP PORT "*_in*" 4 ns
CLKPORT "clk" ;
CLOCK_TO_OUT PORT "*at*" 7 ns CLKPORT "clk" ;
CLOCK_TO_OUT PORT "data_io_{2:4}" 4.5 ns CLKNET "clk_c";
```

Wildcards to Define a Group

```
DEFINE PORT GROUP "data_io_group"
"*io* ;
IOBUF GROUP "data_io_group" IO_TYPE=HSTL15_I ;
```

Wildcards for Timing Preferences In the following example, CELL refers to a register.

```
BLOCK PATH FROM CELL "r2datb*" ;
MULTICYCLE TO CELL "data_out*" 2 X;
MULTICYCLE FROM CELL "r2dat*" TO CELL "equal_out*" 4 X ;
```

Wildcards can also be used with MAXDELAY, although such use is not very common:

```
MAXDELAY NET "*sqmuxaZ*" 1.0 ns ;
```

Wildcards for Clock Nets The following examples show wildcard use for clock nets.

```
BLOCK PATH FROM CLKNET "clk1*" TO CLKNET "clk2*" ;
```

Note

The wildcard expression for clock nets in the BLOCK PATH preference should be specific enough to cover only the paths to be blocked. If "`clk*`" is used, as in:

```
BLOCK PATH FROM CLKNET "clk*" TO CLKNET "clk*"
```

the preference will block all the paths from clk1 to clk1 as well as from clk1 to clk2.

```
MULTICYCLE FROM CLKNET "clki*" TO CLKNET "clkt*" 3 ns;
FREQUENCY NET "clk*" 20 MHz;
USE SECONDARY NET "clk*";
```

Note

PRIMARY NET "CLK*" is not supported.

Although SECONDARY NET "*" can be used, it might produce warning messages.

Other Examples The following examples further illustrate range expansion:

```
// define a group for a 16 bit data bus
```

```
define port group data_out dataout({15:0});
// define a group for the upper/lower 4 bits of the bus
define port group upper_lower dataout({15:12,3:0});
// define a group consisting of only the odd bit positions
define port group odd_bits dataout({15,13,11,9,7,5,3,1}) ;
```

This bus range syntax can be used anywhere a wildcard is legal, not only in the define group command. It also can be applied to wildcard instance names, if you have a “bus” style instance naming convention, where your instance names have a common string prefix and a numeric suffix.

Be careful not to confuse lexical character range specification ({<*from_char*>-<*to_char*>}) with integer range specification ({<*start*>:<*end*>}). The following two examples are NOT equivalent:

`{0-15}` expands to the characters ’0’, ’1’, and ’5’.
`{0:15}` expands to the integers 0 through 15.

See Also ▶“Using Filters and Wildcards” on page 424

- “Preference Syntax Guidelines and Conventions” on page 527
- “Preferences” on page 1195

Using Preference Scripts

Preference scripts enable you to set preferences conditionally and maintain multiple sets of preferences for different device families. A script enables you to define symbols and set specific values using constant expressions for such things as temperatures and frequencies. It also allows you to use arithmetical expressions.

The script commands in the .lpf file are interpreted at map time, and only the appropriate commands are used by map and written into the .prf file. To see how the tools are interpreting the script, examine the .prf file.

When you use a preference script, most editing should be done with a text editor instead of the Diamond preference views. The preference views should be used mainly to examine the design.

Defining Symbols Defined symbols enable you to determine which preferences are to be used in the preference file.

Note

Symbol names are case-sensitive.

Use the @define keyword and a variable name.

Figure 80:

```
@define $<variable_name>;
```

The dollar sign character must prefix the variable name. The variable name can start with any of the following characters:

Figure 81:

```
A-Z a-z $ _ * [ ] / . \ : { } ( ) ?
```

The remaining part of the variable name can contain any of the following characters:

Figure 82:

```
0-9 A-Z a-z , $ _ * [ ] / . \ : { } + - ? ^ ( )
```

Testing for Defined Symbols Use the defined symbols and the @ifdef keyword to determine which preferences are to be used in the preference file:

Figure 83:

```
@ifdef $<variable_name> ;
  <preferences>
@endif ;
```

If the symbol has been defined earlier in the preference file, all preferences between @ifdef and @endif will be considered valid. If the symbol has not been defined, the preferences will be ignored.

Use the @ifndef keyword to test for the nonexistence of a symbol:

Figure 84:

```
@ifndef $<variable_name> ;
  <preferences>
@endif ;
```

This works in the same way as @ifdef, except that the condition is true if the symbol has *not* been defined earlier in the preference file.

Another keyword, @else, can be used after @ifdef or @ifndef to specify an alternate set of preferences when the first condition is not met:

Figure 85:

```
@ifdef $<variable_name> ;
  <preferences>
@else ;
  <alternate_preferences>
@endif ;
```

Tests using @ifdef and @ifndef can be nested to any level.

Examples Using Defined Symbols In the following example, the symbol DEBUG is defined, and it is followed by the @ifdef keyword and a LOCATE preference. Because DEBUG has been defined, the LOCATE preference is used.

Figure 86:

```
@define $DEBUG ;
@ifdef $DEBUG ;
  LOCATE COMP "in" SITE R29C4B ;
@endif ;
```

In this next example, the LOCATE preference is not used, because the DEBUG symbol is not defined:

Figure 87:

```
@ifdef $DEBUG ;
  LOCATE COMP "in" SITE R29C4B;
@endif ;
```

In the following example, the symbol DEBUG is defined. However, the LOCATE preference will not be used, because the test is for DEBUG *not* being defined.

Figure 88:

```
@define $DEBUG;

@ifndef $DEBUG ;
  LOCATE COMP "in" SITE R29C4B ;
@endif ;
```

In the following example, the symbol DEBUG is defined. As as result, the second LOCATE preference after the @else keyword is used, because the @ifndef DEBUG fails.

Figure 89:

```
@define $DEBUG ;

@ifndef $DEBUG ;
  LOCATE COMP "in" SITE R29C4B ;
@else ;
  LOCATE COMP "in" SITE R30C10B ;
@endif ;
```

Example Using Nested Preferences The following example includes a nested FREQUENCY preference. Because both DEBUG and HIGHSPEED are defined, the frequency of the net "clk" will be set to 500.0000 MHz.

Figure 90:

```
@define $DEBUG ;
@define $HIGHSPEED ;

@ifdef $DEBUG ;
    @ifdef $HIGHSPEED ;
        FREQUENCY net "clk" 500.0000 MHz ;
    @else ;
        FREQUENCY NET "clk" 250.0000 MHz ;
    @endif ;
@else ;
    FREQUENCY NET "clk" 250.0000 MHz ;
@endif ;
```

Using Constant Expressions Constant expressions enable you to set a specific value for a defined symbol. You can then test the value using conditional expressions. The same @define keyword is used, but it is associated with a constant value.

Figure 91:

```
@define $<variable_name> <constant_value> ;
```

The constant value can be any integer or floating point number that is within the range supported for numeric values. The value can also be a string that is within the preference naming limitations. The variable can have a string or numeric value, but not both. See "Preference Syntax Guidelines and Conventions" on page 527.

Testing for Constant Expressions You can test a symbol for its value as well as its definition by using the @ifdef and @ifndef keywords:

Figure 92:

```
@ifdef $<constant_compare_expression> ;
  <preferences>
@else ;
  <alternate_preferences>
@endif ;

@ifndef <constant_compare_expression> ;
  <preferences>
@else ;
  <alternate_preferences>
@endif ;
```

The constant compare expression supports a variety of tests:

Figure 93:

```
<constant_compare_expression> :=
  <variable_name> > | >= | = | == | < | <= <constant_value>
```

After defining a symbol with a constant expression, you can use the defined constant in place of a numerical value in the preference by prefixing '$' to the variable name.

Examples Using Constant Expressions In the following example, the symbol MAX_FREQ is defined as 1. The script tests whether MAX_FREQ is equal to 1. Because the test passes, the first FREQUENCY PORT preference is used.

Figure 94:

```
@define $MAX_FREQ 1 ;

@if $MAX_FREQ = 1 ;
  FREQUENCY PORT clk1 200 MHz ;
@else ;
  FREQUENCY PORT clk1 100 MHz ;
@endif ;
```

Variables can also be used to substitute for parameters in preferences. In the following example, the symbols `temp` and `freq` are defined.

Figure 95:

```
@define $temp 100.000000 ;
@define $freq 150.000000 ;
TEMPERATURE $temp C ;
FREQUENCY NET "clk_c" $freq MHz ;
```

After mapping, the .prf file will contain the following preferences:

Figure 96:

```
TEMPERATURE 100.000000 C ;
FREQUENCY NET "clk_c" 150.000000 MHz ;
```

In this next example, BITPATTERN is defined as a string value.

Note

Currently, USERCODE is the only preference that can use a variable defined as a string.

Figure 97:

```
@define $BITPATTERN ""01010011011001010110110101101001" ;
USERCODE BIN $BITPATTERN ;
```

Using Arithmetical Expressions You can use arithmetical expressions with @define, @ifdef, and @ifndef. The usual precedence of operations, from highest to lowest, is supported:

- expressions in parentheses

  Note

  A space must be placed inside each parenthesis.

- multiplication and division (* and /)
- addition and subtraction (+ and -)
- comparison operations (>, <, >=, <=, =)

  The == symbol can also be used with @ifdef and @ifndef to test for equality.

Examples Using Arithmetical Expressions In the following example, A is defined as 10 and B is defined as 4. The @ifdef statement tests whether A plus 4 is greater than 10. Since this is true, the CLOCK_TO_OUT preference is used.

Figure 98:

```
@define $A 10.000000;
@define $B 4.000000;

# TRUE: 14 > 10
@ifdef $A + 4.000000 > 10.000000;
CLOCK_TO_OUT PORT "speaker" 100.000000 ns MIN 100.000000 ns
CLKNET "clk_c" CLKOUT PORT "speaker" ;
@endif;
```

In the following example, the @ifdef statement tests whether A plus B is less than 10. Since this is false, the CLOCK_TO_OUT preference is not used.

Figure 99:

```
@define $A 10.000000;
@define $B 4.000000;

# FALSE: 14 < 10
@ifdef $A + $B < 10.000000;
CLOCK_TO_OUT PORT "speaker" 900.000000 ns MIN 100.000000 ns
CLKNET "clk_c" CLKOUT PORT "speaker" ;
@endif;
```

In this next example, the comparison in the @ifdef statement utilizes parentheses that change the precedence so that 2 plus 4 is calculated first. Notice that a space is placed inside each parenthesis so that the expression will be parsed correctly. In the end, the expression is true, so the CLOCK_TO_OUT preference is used.

Figure 100:

```
@define $A 10.000000;
@define $B 4.000000;

# TRUE: ( ( 6.000000 ) * 10.000000) / 2 > 25, 30 > 25
@ifdef ( ( 2.000000 + $B ) * $A ) / 2.000000 > 25.000000;
CLOCK_TO_OUT PORT "speaker" 3000.000000 ns MIN 100.000000 ns
CLKNET "clk_c" CLKOUT PORT "speaker" ;
@endif;
```

The following example adds two new variables, G and H. The G variable is used in a comparison with 20, which is true, so the CLOCK_TO_OUT preference is used.

Figure 101:

```
@define $A 10.000000;
@define $B 4.000000;
@define $G $A + 10;
@define $H $G - $B;

@ifdef $G == 20;
   CLOCK_TO_OUT PORT "speaker" 900.000000 ns MIN 100.000000 ns
CLKNET "clk_c" CLKOUT PORT "speaker" ;
@endif;
```

In this next example, H, which computes to 16, is used in a comparison with 20. Since H is less than 20, the CLOCK_TO_OUT preference is not used.

Figure 102:

```
@define $A 10.000000;
@define $B 4.000000;
@define $G $A + 10;
@define $H $G - $B;

@ifdef $H > 20;
   CLOCK_TO_OUT PORT "speaker" 1000.000000 ns MIN 100.000000 ns
CLKNET "clk_c" CLKOUT PORT "speaker" ;
@endif;
```

See Also ▶"Preferences" on page 1195

Generating Preferences from Synthesis Tools

Diamond enables you to add preferences generated from SDC or FDC synthesis tools to the logical preference file (.lpf) for the design. With the Output Preference File strategy option turned on, a synthesis .lpf file gets generated when you run Translate Design or when you run Synthesize Design for LSE. You can then add these preferences to the design's .lpf file and modify them as desired.

To generate the synthesis preference file:

1. In the File List view, select the strategy you want to use and make it the active strategy.
2. Choose **Project > Active Strategy > <synthesis tool> Settings**
3. Scroll down to the **Output Preference File** and double-click the cell in the Value column.
4. Select **True** from the drop-down menu and click **OK**.
5. In the Process view, do one of the following:
   - Double-click **Translate Design** if your synthesis tool is Synplify Pro or Precision.
   - Double-click **Synthesize Design** if your synthesis tool is LSE.

   Diamond synthesizes the design and outputs the preferences to the synthesis .lpf file.

The synthesis .lpf file is located in the <project_name><project_name> folder. The .lpf file name includes the project name, followed by the name of the synthesis tool; for example, attributes_attributes_synplify.lpf.

The following is an example of an .lpf file generated by Synplify:

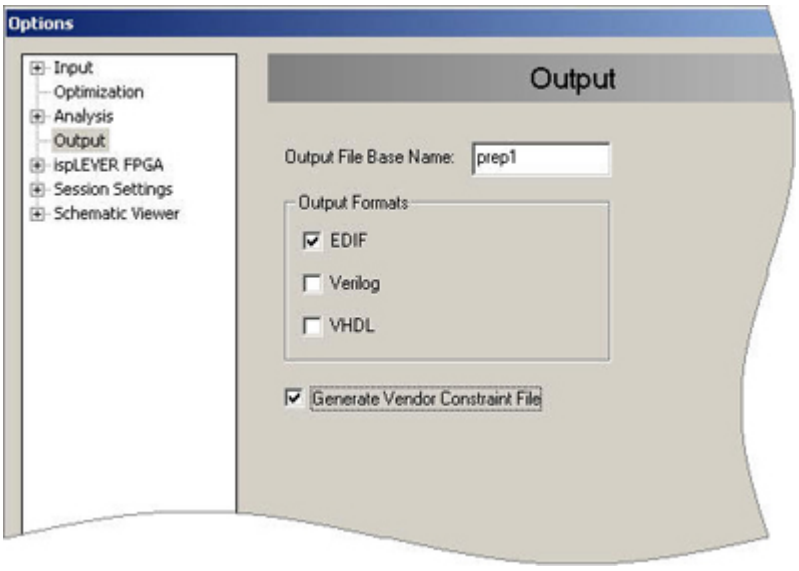
```
#
# Logical Preferences generated for Lattice by Synplify 9.0.0,
Build 260R.
#
# Period Constraints
FREQUENCY PORT "CLK" 200.0 MHz;
# Output Constraints
CLOCK_TO_OUT "Q_0" 5 NS CLKPORT = "CLK";
CLOCK_TO_OUT "Q_1" 5 NS CLKPORT = "CLK";
CLOCK_TO_OUT "Q_2" 5 NS CLKPORT = "CLK";
CLOCK_TO_OUT "Q_3" 5 NS CLKPORT = "CLK";
CLOCK_TO_OUT "Q_4" 5 NS CLKPORT = "CLK";
CLOCK_TO_OUT "Q_5" 5 NS CLKPORT = "CLK";
CLOCK_TO_OUT "Q_6" 5 NS CLKPORT = "CLK";
CLOCK_TO_OUT "Q_7" 5 NS CLKPORT = "CLK";
# Input Constraints
INPUT_SETUP "S_L" 5 NS CLKPORT = "CLK";
INPUT_SETUP "S1" 5 NS CLKPORT = "CLK";
INPUT_SETUP "S0" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_0" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_1" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_2" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_3" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_4" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_5" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_6" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d0_7" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_0" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_1" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_2" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_3" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_4" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_5" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_6" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d1_7" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_0" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_1" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_2" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_3" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_4" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_5" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_6" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d2_7" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_0" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_1" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_2" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_3" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_4" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_5" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_6" 5 NS CLKPORT = "CLK";
INPUT_SETUP "d3_7" 5 NS CLKPORT = "CLK";
BLOCK ASYNCPATHS;
# End of generated Logical Preferences.
```

How to Turn on Preference Output for Precision The Mentor Graphics Precision synthesis tool provides an option that outputs a constraint file for Diamond. When you select this option, some of the synthesis constraints will be written to the design's .prf file. You can then copy these preferences to the logical preference file (.lpf).

To turn on Generate Constraint File in Precision:

1. Open your <*project_name*>.psp file in Precision.
2. Choose **Tools > Set Options**.
3. In the left pane of the dialog box, select **Output**.

Figure 103:



4. In the Output pane, select **Generate Vendor Constraint File** and click **OK**.
5. In Diamond, select the target device, and then double-click **Translate Design**.

   Diamond synthesizes the design and outputs the Precision preferences to the .prf file.

To write the synthesis preferences to the active .lpf file:

1. Using a text editor, open the generated synthesis .lpf file and also the desired .lpf file for the design.
2. Copy the preferences from the synthesis .lpf file, paste them into the .lpf file for the design, and then close the synthesis .lpf file.
3. In the .lpf file for the design, choose **File > Save**.

   When the modified .lpf file is set as the active one, Diamond will use these added preferences in processing the design.

See Also ▶"Integrated Synthesis" on page 583

Saving Preferences

Each time you issue a process command, such as Map or Place & Route, Diamond prompts you to save any new modifications to your project files, including the logical preference file (.lpf) for the current implementation. Diamond also prompts you to save new or modified preferences when you close the project or exit the software.

You will not be prompted, however, to save changes each time you close a preference-editing view. Your preference modifications are kept in memory as long as Diamond is running. An asterisk on the tab of any preference view indicates unsaved preference changes, and so does the "Preferences Modified" indicator on the status bar. The "Preferences Modified" statement remains on the status bar as long as there are preference changes in memory, even when all preference views are closed.

You can save preference changes to the .lpf file at any time from a preference-editing view. Click one of the available Save buttons on the toolbar or use any of the available Save commands from the File menu.

Note

When you use any of the Save commands after running Place & Route, the design gets initialized back to pre-map status.

Save <file_name>.lpf Saves preference changes to the .lpf file for the active implementation. The Save command is available from any preference view and saves all preference changes—not just those that were made from the active preference view.

Save <file_name>.lpf as Saves preference changes to a different .lpf file name or to a different directory that you specify. When you use the "Save as" command and give the .lpf file a new name, the new .lpf file is automatically added to the Constraints Files folder and set as the active constraint file for the current implementation.

Save All Saves preference changes to the currently active .lpf file, as well as changes that were made using other tools. For example, if changes are made using Spreadsheet View and Power Calculator, it will save the Spreadsheet View changes to the .lpf file, and it will save the Power Calculator changes to the .pcf file. At least one of the preference-editing views must be open in order to save preference changes with the Save All command.

See Also ▶"Preference-Editing Views and Memory" on page 407

Previewing Preferences Before Saving

You can quickly preview all preferences in a read-only file to ensure that preference changes are correct before you save them. The Preference Preview lists the preferences from the Logical Preference File (.lpf), plus all preference changes that exist in memory only.

Preference Preview is also useful for copying in-memory preferences and pasting them into a different .lpf file. This allows you to save the in-memory changes without changing the currently active preference file.

To preview all preferences:

- Choose **View >** **Preference Preview**.

  A read-only preference file opens in a separate view.

  Note

  The Preference Preview does not list unsaved changes that were made manually to the .lpf file.

See Also "Preference-Editing Views and Memory" on page 407

Saving Preferences to the Currently Active .lpf File

You can immediately save preference changes to the currently active .lpf file from any preference-editing view.

To save preference changes to the currently active .lpf file:

- From a preference-editing view, choose **File >** **Save <file_name>.lpf**.

  Diamond saves your changes to the active logical preference file and project directory. The design is initialized back to pre-map status.

See Also "Preference-Editing Views and Memory" on page 407

Saving Preferences to a Different .lpf File

The "Save as" feature is useful for creating a set of modified preferences to use with an existing implementation or a new one. When you use the "Save as" command, Diamond adds the new .lpf file to the Constraint Files folder and sets it as the active constraint file for the current implementation. If you have already placed and routed the design, the "Save as" command will initialize the design back to pre-map status.

To save preferences to a new .lpf file:

1. Activate Spreadsheet View, Package View, Device View, or Floorplan View; or double-click the .lpf file to open it in the Text Editor window.
2. Choose **File > Save <file_name>.lpf as**.
3. In the dialog box, type the desired name for the .lpf file in the Name text box, navigate to the desired directory, and click **Save**.

   The new .lpf file is added to the Constraint Files folder and is set as the active one for the current implementation. The design is initialized back to pre-map status.

To revert to the previous .lpf file for the current implementation:

- In the Constraint Files folder, right-click the previous .lpf file and choose **Set as Active Preference File**.

See Also "Preference-Editing Views and Memory" on page 407

Saving Preferences and Other Changes

The Save All command saves all modifications made to your project files. It saves preference changes to the .lpf source file, and it saves changes made with other tools to source files or analysis files.

To save all changes:

1. Make sure that at least one preference-editing view is open in Diamond.
2. From any view, choose **File > Save All**.

Applying External Preference Changes

While using the preference views, you may sometimes need to modify preferences externally. For example, you might want to open the logical preference file (.lpf) in an external text editor to quickly modify a UGROUP or REGION. You can afterwards view the changes in Diamond before processing the design.

Before applying external preference changes, make sure that there are no preference changes in memory. If there are unsaved preference changes in Diamond, save them to a different .lpf file or close and reopen Diamond without saving the changes.

Caution

When there are preference changes in memory, using the Save command in a preference view will cause you to lose the external preference changes.

External changes made to the .lpf file will be reflected in the preference views when you reopen them. If a preference view is already open when the

changes are made externally, the changes might not be displayed. To display the external changes, close and reopen any preference view.

Rerun the Map Design process to apply the changes.

Discarding In-Memory Preference Changes

There are two ways to delete all modified preferences that are in memory.

Discard In-Memory Preference Changes Only The “Discard Modified Preferences” command clears all unsaved preferences from memory and displays only those that are in the logical preference file.

To discard in-memory preferences:

1. Choose **File > Discard Modified Preferences**.
2. In the warning dialog box, click **Yes** to continue.

   The unsaved preferences are cleared from all open preference views. Diamond removes the asterisks from the preference view tabs and the “Preferences Modified” indicator from the status bar.

Discard All In-Memory Design Changes The “Clear Tool Memory” command discards all in-memory design changes, including preferences, and closes the open tools. Before the memory is cleared, a Confirm dialog box enables you to save the changes to the appropriate source file. When you select “Deselect All,” Diamond discards all changes.

To clear tool memory:

1. Choose **Tools > Clear Tool Memory**.

   The Confirm dialog box opens and lists in the upper section the tools that have unsaved changes. The lower section lists the affected files, including the .lpf file.
2. In the Confirm dialog box, clear the check mark in front of the .lpf file or click **Deselect All**, and then click **OK**.

   Diamond closes the open tools and removes the changes from memory.

Checking Design Rules

Diamond provides PIO design rule checking (DRC), which enables you to validate I/O placement, logic, and routing.

During load time and initialization, Diamond’s preference views interpret the database and preference files in a similar manner as the Map and Place & Route processes. Syntax or semantic violations within the logical preference file are also treated in a manner similar to the Map and Place & Route

processes. Illegal preferences are ignored but retained in the logical preference file and the physical preference file. Diamond loads and applies the legal subset it detects and displays error and warning messages.

After the project is loaded, Diamond provides access to on-demand and real-time PIO DRC Check. All DRC reports are presented in the Output window for the current session.

For LatticeSC, LatticeECP3, and MachXO2 devices, a PIO DRC report is placed in the Tool Reports folder when you run on-demand PIO DRC. Any errors or warnings appear in the top part of the Summary section of the report, and each error or warning is hyper-linked to the appropriate DRC rule in the Details section. See "PIO DRC Rules" on page 552.

See Also ▶"PIO DRC" on page 550

PIO DRC

PIO design rule check can catch many common errors and prevent wrong pin usage at the early stage of the design process. It helps you ensure that design I/Os are assigned correctly before the design is mapped, placed, and routed. This can dramatically reduce design errors, help you meet timing requirements, optimize for both the FPGA and printed circuit design, and reduce the number of design iterations.

PIO DRC performs logical and physical tests to validate I/O placement such as signal standard and VCCIO combinations bank by bank, true and emulated LVDS combinations, VREF requirements, and potential conflicts with user assignments made to multi-function pins like those of the sysCONFIG interface. It is available as real-time and on-demand DRC.

Real-time PIO DRC Real-time PIO DRC, when enabled, runs automatically in real time for items that have been modified. For example, when you import constraints from an external source, real-time PIO DRC runs automatically. Real-time PIO DRC also runs automatically when you set I/O preferences through one of the preference-editing views such as Spreadsheet View.

You can enable or disable real-time PIO DRC in Spreadsheet View. The second  button on the toolbar acts as a toggle key that turns real-time PIO DRC on or off. By default, it is turned on.

On-demand PIO DRC On-demand PIO DRC means that you can run it manually at any time for the whole design. For example, when you have modified preferences in Spreadsheet View, you can use the Preference PIO DRC Check  command to find out if any PIO design rules have been violated in the entire design as a result of the modifications. On-demand PIO DRC is available in Spreadsheet View, Package View, Device View, and Floorplan View.

See Also ▶"Performing On-Demand PIO Design Rule Check" on page 551

Performing On-Demand PIO Design Rule Check

The PIO DRC feature validates pin assignments to banks based on legal combinations of signal standards, including true and emulated LVDS combinations. Diamond performs immediate real-time PIO DRC on any preference changes. But you can run PIO DRC on the entire design, after modifying preferences, by running on-demand PIO DRC.

To perform PIO design rule checking:

1. From Diamond's Tools menu, select one of the following views:
   - Spreadsheet View
   - Package View
   - Device View
   - Floorplan View
2. Choose **Design > Preference PIO DRC**.

   Diamond checks the I/Os . A message box reports whether errors or warnings were encountered. A log of errors and warnings appears in the Output window.

   If your design's target device is LatticeSC, LatticeECP3, or MachXO2, a DRC report will appear in the Tool Reports folder. Any DRC errors or warnings are listed at the top of the Summary section. Click the hyperlinks to jump to the design rule description in the Details section.

See Also "PIO DRC" on page 550

"PIO DRC Rules" on page 552

Post-Map DRC Check

In the post-map stage, DRC performs logical and physical tests: Block, Net, Pad, Clock Buffer, Name, and Primitive Pin. Reported errors from DRC check indicate conditions where routing or component logic will not operate correctly. Warnings indicate conditions where routing or logic is incomplete or the condition is incorrect but not serious. Some conditions considered warnings might be reported as errors when DRC is performed by the Generate Bitstream Data (bitgen) process.

Post-Map DRC Check is available in the post-map or post-PAR stage in Spreadsheet View, Package View, Device View, and Floorplan View.

See Also "Performing On-Demand PIO Design Rule Check" on page 551

PIO DRC Rules

The following tables describe the PIO design rules for LatticeSC, LatticeECP3, and MachXO2 device families. The tables are organized by the category of elements affected by the PIO design rules: port, pin, bank, device, Vref, IOBUF, and programmable interface cells (PIC). Example messages are provided, as well as suggestions for resolving errors and warnings.

Table 39: Port-Related Rules

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
NSidePortAssignmentCheck N-side port assignment check	error	LatticeSC LatticeECP3 MachXO2	The n-side port [PORT_NAME] assignment is not supported.	Use p-side port to make the assignment.
PortBusDirectionConsistency Bus direction consistency check	warning	LatticeSC LatticeECP3 MachXO2	The bus [BUS_NAME] has members with different port direction.	Use the same direction for the bus.
PortBusIoTypeConsistency Bus IO Type consistency check	warning	LatticeSC LatticeECP3 MachXO2	The bus [BUS_NAME] has members with different IO types [IO_TYPE, IO_TYPE,…].	Use a consistent IO Type for all the members in the bus.
PortClockAssignment[1] Clock port assignment check	warning	LatticeSC LatticeECP3 MachXO2	The clock port [PORT_NAME] is assigned to a non-clock-dedicated pin [PIN_NAME]	Use dedicated clock resources for the port.
PortConnection Port connection check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] doesn't have a load.	Connect or remove the port.
PortDifferentialIoType Differential port IO Type check	error	LatticeSC LatticeECP3 MachXO2	Non-differential IO type [IO_TYPE] is assigned to the differential port [PORT_NAME].	Use a differential IO type for the port.
PortDifferentialPSideExistence Differential port positive side existence check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] doesn't have a complementary positive side port defined.	Check port definition.
PortExistence Port existence check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] doesn't exist in the design.	Remove the port.
PortFloatingNumber Floating port number check	error	LatticeSC LatticeECP3 MachXO2	The number of unassigned ports ([NUMBER]) exceeds the number of available pins [NUMBER] left based on the current assignment.	▶ Rearrange pin assignments. ▶ Change to a large device with more pins.

Table 39: Port-Related Rules (Continued)

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
PortGlobalClkNumber Global clock total number check	warning	LatticeSC LatticeECP3 MachXO2	The number of clocks [NUMBER] exceeds the available number of global clock dedicated pins [NUMBER].	Warning: non-clock-dedicated pins will be used for some clock ports.
PortGroupExistence Port group existence check	error	LatticeSC LatticeECP3 MachXO2	The port group [GROUP_NAME] is not defined.	Remove the port group.
PortGroupUniqueness Port group uniqueness check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME….] defined in group [GROUP_NAME] has been included in multiple port groups.	Change group definitions and make sure that the port is included in only one port group.
PortInGroupExistence Port defined in PORT GROUP existence check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] defined in port group [PORT GROUP] does not exist in the design.	Remove the port.
PortPllAssignment[2] PLL ports assignment check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] connects to a PLL but is not assigned to a PLL dedicated pin.	Assign the port to a PLL dedicated pin.
PortTotalDifferentialPairNumber Differential port total number check	error	LatticeSC LatticeECP3 MachXO2	The total number of differential ports ([NUMBER]) exceeds the total differential pin pairs available [NUMBER] in the target device [DEVICE_NAME].	▶ Change to a device with more differential pin pairs. ▶ Migrate to a large device with more differential pin pairs.
PortTotalIoTypeNumber IO type number check	error	LatticeSC LatticeECP3 LatticeECP4 MachXO2	The number of VCCIO or banks required exceeds the total number of banks ([NUMBER]) available based on the current assignment.	▶ Reduce the number of IO Types. ▶ Use compatible IO types. ▶ Change to a large device with more banks.
PortTotalNumber Port total number check	error	LatticeSC LatticeECP3 MachXO2	The total number of ports ([NUMBER]) exceeds the total assignable pins available ([NUMBER]) in the target device [DEVICE_NAME].	Change to a device with more available pins.
PortVrefAssignment Port assigned to a vref which is in different bank	error	LatticeSC LatticeECP3 MachXO2	The port [PORT NAME] assigned to bank [BANK#] is also assigned to VREF [VREF name], which is in different bank [BANK#].	Assign the port to the same bank [BANK#] of the VREF [VREF NAME].
PortVrefSupport Port Vref support check	error	LatticeSC LatticeECP3 MachXO2	The Port [PORT_NAME] does not support Vref.	Change IOBUF setting to enable VREF.

Notes

[1] The PortClockAssignment rule checks a “CLOCK Input” type of port (a clock port) and ensures that it is assigned to a PCLK or PLL pin. In order to achieve optimized performance for some cases, it is suggested that the CLOCK input be connected to a dedicated PCLK pin or PLL pin. The PIO DRC will issue a Warning if the clock is not connected to one of these pins. For high speed designs, it is recommend that a PCLK pin be used for going directly to the Primary or Edge clock tree and a PLL pin be used if the clock is going to a PLL. If speed is not a requirement for the design, these Warning messages can be ignored.

[2] The PortPllAssignment rule checks a port that connects to a PLL (a PLL port) and ensures that it connects to a PCLK pin or PLL pin. In some cases, such as the DDR edge aligned interface where the high speed clock is going directly to a PLL, the CLOCK port must connect to a PLL pin. In this case, if the port is assigned to a PCLK pin, PIO DRC will not issue a warning. When this is the case, make sure that the PLL port is assigned to a PLL pin.

Table 40: Pin-Related Rules

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
PinDDRSite DDR site assignment check	error	LatticeECP3	Cannot place PIO comp [COMP NAME] on the PIO site [SITE VALUE] because the types of their IOLOGICs are incompatible.	Assign it to a supported site which is on the left or right hand side.
PinDdrUsageCheck Pin DDR usage pattern check	error	LatticeECP3	The following IDDRX(s) [port_names] and ODDRX(s) [port_names] can not be placed in the same DQS group.	Check these ports' DDR usage, and follow the DDR usage model.
PinDifferentialCapability Pin differential capability check	error	LatticeSC LatticeECP3 MachXO2	The differential port [PORT_NAME] is assigned to a non differential pin [PIN_NAME] (or illegal differential pair [NAME]).	Reassign the port to a valid differential pin.
PinDifferentialPairComplementary Differential pin pair usage check	error	LatticeECP3	The differential port [port_name] is assigned to the pin [pin_name], but this pin's complementary pin [pin_name] is taken by the port [port_name].	Assign the port to a different differential pin pair, or release the complementary pin.
PinDifferentialPairTrue Differential pin pair positive pin check	error	LatticeECP3	The differential port [port_name] must be placed on a positive pin of a differential pin pair.	Assign the port to a positive pin of a differential pin pair.
PinDifferentialPSideExistence Differential pin pair positive pin existence check	error	LatticeSC LatticeECP3 MachXO2	The differential port [PORT_NAME] cannot be assigned to the pin [PIN_NAME] because this pin doesn't have a complementary positive pin.	Assign the port to a valid differential pin pair.
PinDirection Directional pin assignment check	error	LatticeSC LatticeECP3	The [input/output] port [PORT_NAME] is assigned to an [output/input] dedicated pin [PIN_NAME].	Reassign the port to a valid pin.

Table 40: Pin-Related Rules (Continued)

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
PinExistence Pin existence check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] is assigned to a nonexistent pin [PIN_NAME].	Reassign the port to a valid pin.
PinFeatureSupport Pin-supported features check	error	LatticeSC	The port [port_name] with [MODE_NAME] mode uses [VCCIO3.3|LVDS|RSDS|HYPT that is not supported by the pin [pin_name].	Assign the port to a pin that supports the VCCIO/LVDS/RSDS/HYPT feature.
PinInputOnly Pin input only check	error	MachXO2	The [output|BIDI] port [port_name] is placed on an input-only pin [pin_name].	Assign the port to a different pin that supports output mode.
PinIoTypeCapability Pin IO Type capability check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] has IO Type [IO_TYPE] assigned, but the IO Type is not supported by the assigned pin [PIN_NAME].	Use a pin that supports the IO Type.
PinStateValue Pin state value check	error	LatticeSC LatticeECP3 MachXO2	The pin [PIN_NUMBER] is [reserved/assigned to [PORT_NAME]] and cannot be assigned to the port [PORT_NAME].	Change the pin's assignment.
PinSysConfig SysConfig pin assignment check	error	LatticeSC LatticeECP3 MachXO2	The state of the assigned pin [PIN_NAME] conflicts with the config mode and cannot be assigned to the port [PORT_NAME].	▶ Change pin assignment ▶ Change SysConfig setting.
PinTypeValue Pin type value check	error	LatticeSC LatticeECP3 MachXO2	The type value [PIN_TYPE] of pin [PIN_NAME] is invalid and cannot be assigned to the port [PORT_NAME].	Change the pin's assignment.
PinVrefAvailability VREF pin availability check	error	LatticeSC LatticeECP3 MachXO2	VREF is required in bank [BANK#] but there is no VREF pin available in the bank.	Release a [VREF pin (LatticeSC/LatticeECP3) / pin for VREF(MachXO2)] in bank [BANK#].
PinVrefGroup Pin Vref group check	error	LatticeSC LatticeECP3 MachXO2	Pins in the VREF group [GROUP_NAME] require different VREF values.	Make sure that all the pins in the same VREF group require the same VREF value.

Table 41: Bank-Related Rules

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
BankDifferentialLimit Bank differential output limitation check	error	LatticeSC	HYPT buffer is used together with LVDS/RSDS in the same bank. The port [port_name] is in conflict with other ports in the bank [bank_value] (refer to port [port_name]).	Assign the port t o a different bank.
BankDiffPairCapacity Bank differential pin pair capacity check	error	LatticeSC LatticeECP3 MachXO2	The number of differential ports ([NUMBER]) assigned to the bank [BANK_NAME] exceeds the number of differential pin pairs ([NUMBER]) available in the bank.	Use multiple banks.
BankExistence Bank existence check	error	LatticeSC LatticeECP3 MachXO2	Bank [BANK] is not available in the target device [DEVICE_NAME].	Use valid bank number.
BankIoTypeCapability Bank IO Type capability check	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] has IO Type [IO_TYPE] assigned, but the IO Type is not available in the assigned bank [BANK_NAME].	Use a bank that supports the IO Type.
BankIoTypeCompatability Bank IO Type compatibility check	error	LatticeSC LatticeECP3 MachXO2	Ports [PORT_NAME, PORT_NAME...] with incompatible IO types [IO_TYPE, IO_TYPE,…] are assigned to the same bank [BANK_NAME].	Use multiple banks.
BankPinCapacity Bank pin capacity check	error	LatticeSC LatticeECP3 MachXO2	The number of ports ([NUMBER]) assigned to the bank [BANK_NAME] exceeds the number ([NUMBER]) of pins available in the bank.	Use multiple banks.
BankUniqueINRD BANK unique INRD check	error	MachXO2	In the bank [bank_number], more than one INRD signals [signal_name, signal_name,…] are found for the ports [port_name, port_name, ...]	Assign the port to a bank with the same INRD signal.
BankUniqueLVDS BANK unique LVDS check	error	MachXO2	In the bank [bank_number],, more than one LVDS signals [signal_name, signal_name,…] are found for the ports [port_name, port_name, ...]	Assign the port to a bank with the same LVDS signal.

Table 41: Bank-Related Rules (Continued)

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
BankUniquePG BANK unique PG check	error	MachXO2	In the bank [bank_number],more than one PG signals [signal_name, signal_name,…] are found for the ports [port_name, port_name, ...]	Assign the port to a bank with same PG signal
BankVccioValueConsistency Bank VCCIO value consistency check	error	LatticeECP3 MachXO2	In the bank [bank_value], VCCIO for [port_name] is [VCCIO_value] and is incompatible with other ports that use VCCIO [VCCIO_value] (refer to port [port_name])	Assign the port to a compatible bank.
BankVcmtConsistency Bank VCMT usage and IO_TYPE consistency check	error	LatticeSC	The ports [port_name] and [port_name] in the bank [bank_value] use VCMTs but their IO_TYPEs [IO_TYPE_value] and [IO_TYPE_value] are different.	Assign the port to a different bank, or modify IO_TYPE for one of the ports.
BankVrefCapacity Bank VREF capacity check	error	LatticeSC LatticeECP3 MachXO2	The number of VREF required ([NUMBER]) exceeds the number of VREF capacity ([NUMBER]) supported by the bank/device [BANK_NAME/ DEVICE_NAME].	Reduce the number of different VREF required for [{BANK/DEVICE}] [{BANK_NAME}/ {DEVICE_NAME}].
BankVrefLoad1DriverMatch Bank VREF1 load driver consistency check	error	LatticeSC	The port [port_name] in the bank [bank_value] uses VREF1_LOAD, but there is no VREF1_DRIVER in that bank.	Add VREF1_DRIVER to the bank.
BankVrefLoad2DriverMatch Bank VREF2 load driver consistency check	error	LatticeSC	The port [port_name] in the bank [bank_value] uses VREF2_LOAD, but there is no VREF2_DRIVER in that bank.	Assign the port to a different bank.
BankVrefLoadConsistency Bank VREF load IO_TYPE consistency check	error	LatticeSC	In the bank [bank_value], VREF for [port_name] is different from the other two VREFs for ports [port_name] and [port_name]	Assign the port to a compatible bank.
BankVttDdrIIVcmtExclusive VTT/DDR_II and VCMT exclusive check	error	LatticeSC	VCMT buffer is used together with VTT/DDR_II in the same bank. The port [port_name] is in conflict with other ports in bank [bank_value] (refer to port [port_name]).	Assign the port t o a different bank.

Table 41: Bank-Related Rules (Continued)

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
BankVttValueConsistency Bank VTT value consistency check	error	LatticeSC LatticeECP3	In the bank [bank_value], VTT for [bank_port] is [VTT_value] and is incompatible with other ports that use VTT [VTT_value] (refer to port [port_name])	Assign the port to a compatible bank.
ConfigBankIOVoltageMismatch PERSISTENT Config Bank I/O voltage check	error	LatticeSC LatticeECP3 MachXO2	Cannot assign [PORT NAME] to [IO_TYPE] in bank [BANK#] because PERSISTENT is turned ON and CONFIG_IOVOLTAGE do not match the I/O voltage of the [PORT NAME] in bank [BANK#] .	Either set PERSISTENT to OFF or match CONFIG_IOVOLTAGE to an acceptable [IO_TYPE] for the port [PORT NAME].

Table 42: Device-Related Rules

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
DeviceVccioCapacity Device VCCIO capacity check	error	LatticeSC LatticeECP3 MachXO2	The number of VCCIO or banks required exceeds the total number of banks available ([NUMBER]) based on the current assignment.	Use large devices.
DeviceVrefCapacity Device VREF capacity check	error	LatticeSC LatticeECP3 LatticeECP4 MachXO2	The number of VREF required ([NUMBER]) exceeds the number of VREF capacity ([NUMBER]) supported by the device [DEVICE_NAME].	Use large devices with more VREF pins.

Table 43: Vref Group-Related Rules

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
VrefGroupExistence Vref group existency check	error	LatticeSC LatticeECP3 MachXO2	VREFGROUP [GROUP NAME] assigned to the port [PORT NAME] is not defined	Define the Vref Group.

Table 44: IOBUF-Related Rules

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
IOBUFAttributeNameValidation IOBUF attribute name validation	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] has invalid IOBUF attribute name [KEY_NAME].	Check the string [KEY_NAME] for typos, or use a valid attribute name.
IOBUFAttributeValueValidation IOBUF attribute value validation	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] has some invalid IOBUF attribute values: [KEY: VALUE], [KEY:VALUE] ….	Check the assignments for those attribute and use valid values.
IOBUFLegalCombinationTableValidation IOBUF legal combination table validation	error	LatticeSC LatticeECP3 MachXO2	The port [PORT_NAME] has invalid combination of the IOBUF settings according to the device I/O specifications.	Check the IOBUF setting and correct the errors.

Table 45: PIC-Related Rules

Rule Name and Description	Severity	Affected Devices	Message Example	Suggestion
PicCompatibleOutput PIC IOLOGIC output and SHX4/DDR4 compatible check	error	LatticeSC	The port [port_name] at pad [A|B|C|D] cannot be used as output (or input) when the [port_name] at pad [A|B|C|D] is in SHX4/DDRX4 mode (or using AIL).	Assign the port to a different PIC (A group of PIO pins consists of adjacent A/B/C/D pads).
PicCompatibleX24AilTri PIC IOLOGIC x2/x4/Ail/Tri-State compatible check	error	LatticeSC	The port [port_name] at pad [A|B|C|D] and the port [port_name] at pad [A|B|C|D] are competing for DDR resources (x2,x4,AIL or tri-state), they cannot be shared in one PIC.	Assign the port to a different PIC (A group of PIO pins consists of adjacent A/B/C/D pads).

Analyzing SSO

For many FPGA devices, Diamond enables you to run an analysis of simultaneous switching outputs (SSO). SSO analysis describes the noise on signals caused by a large number of out drivers that are switching at the same time. When multiple output drivers change state at the same time, the changing current in the power system induces a voltage that results in power supply disturbances. These disturbances, or noise, can cause undesired transient behavior among output drivers, input receivers, or internal logic. Analysis of simultaneous switching outputs helps ensure that your I/O plan meets the I/O standards and power integrity requirements of the PCB design.

SSO Calculator The SSO calculator estimates Simultaneous Switching Noise (SSN) affecting a victim pin according to the switching characteristics of aggressor pins. It does this either on a victim-pin-by-victim-pin basis (Pin

Analysis) or on the basis of the worst-case victim pin placement in each bank (Bank Analysis). In many cases, adjusting the location of aggressor pins relative to victims can help mitigate noisy conditions. Noise is characterized by the calculator as ground bounce or voltage drop depending on the type of analysis run. The calculator will produce worst-case bank-based ground bounce results for those banks that have partial signal assignments, and it will produce a more accurate pin-based ground bounce and voltage drop results if all signals of a bank are assigned.

The following calculation is made for a victim pin (in Bank-based Analysis, the worst case victim is used for each bank):

Pass/Fail status: Fails if (Total Noise) > (Pass/Fail criteria) * ALLOWANCE

where:

Total Noise (mV) = Board Noise + Device Noise

Board PCB Noise can be on the ground plane, for ground bounce calculations; or it can be on the power plane, for Vcc drop calculations. PCB estimated ground plane noise is entered as +X mV. PCB estimated power plane noise is entered as -X mV.

Device noise is the maximum noise imposed on a victim pin by each of the aggressor pin groups. Aggressor groups are specified by using SwitchingID assignments. Reports show which aggressor group is causing the most noise on a victim.

The Pass/Fail criteria is based on the IO_TYPE of the victim. SSO allowance is a user override that adjusts the pass/fail criteria, where default is no change to the pass/fail criteria (100%)

SSN Accuracy SSO calculator estimations are based off of hardware data collected during characterization. The measurements were taken using Lattice test PC boards with signal loading on individual banks. Because the total SSN is a function of PCB ground and board inductance, care must be given to apply accurate SSO preferences based on the parasitic inductances of your system environment.

About Pin-Based Analysis Reports Pin-based analysis produces estimates for ground bounce and voltage drop SSN per pin. Pin-based analysis is performed after design signals have been assigned to pins. These pin assignments can be made by setting preferences or by placing and routing the design. The analysis calculates the noise that each pin, as a victim, experiences. It reports the pass/fail status of each pin and where the noise originates—the aggressor group identified by SwitchingID.

About Bank-Based Analysis Reports Bank-based analysis produces ground bounce SSN per bank. When at least one signal is assigned only to a bank and not to a pin, bank-based calculation is performed. Compared to pin-based analysis, bank-based analysis assumes worst case placement in the bank for the victim and set of aggressors. Any placement information that does exist is ignored, and worst-case placement is used.

For both Pin-based and Bank-based analysis, pins or banks programmed for signal standards (IO_TYPE) other than those that the calculator supports are ignored.

Addressing SSN Problems The SSO calculator is a good tool for confirming the quality of an I/O plan. After legal signal assignments have been verified and placed using the I/O Assistant strategy, use the SSO calculator to adjust relative placement where SSN margins are exceeded.

The following adjustments to your I/O plan can help avoid SSN problems:

- Spread aggressors away from victim pins. Use the “Assign Pin” feature of the Assign Pins dialog box in Spreadsheet View to stagger package pin assignments.
- Use slow slew outputs.
- Use lower drive strengths

See Also
- “Running SSO Analysis” on page 561
- “Viewing SSO Analysis Results” on page 562
- “Exporting an SSO Report in CSV Format” on page 563

Running SSO Analysis

For bank-based SSO analysis, the SSO calculator requires a SwitchingID that indicates the output drivers that will be switching in parallel, and the banks to which the drivers will be assigned. For pin-based analysis, the SSO calculator also requires detailed signal-to-pin placement. You can specify these parameters in the Port Assignments sheet of Spreadsheet View at any stage of the design flow. The more complete your assignments, the more accurate the analysis results will be.

Because the SSO calculator uses the in-memory state of preferences, you are not required to save the SSO parameters in order to run an analysis. This enables you to experiment with placement and other parameters before committing a change to the logical preference file (.lpf).

To run SSO analysis:

1. In Spreadsheet View, select the Port Assignments tab.
2. Assign pin locations or just the bank locations to the output or bidirectional signals that you wish to analyze.
3. Select from the following attributes for each signal to be analyzed:
   - an appropriate I/O buffer from the IO_TYPE column
   - a drive strength for the output buffers from the DRIVE column
   - a slew rate from the SLEWRATE column
4. Type a capacitive output load (pF) value in the Outload column.

5. Specify settings in the following SSO columns for each signal to be analyzed:
   - SwitchingID – Type an alphanumeric value to identify the aggressor output drivers that can switch at the same time.

     If you do not specify a Switching ID, the Default_ID will be assigned to all output pins.
   - Ground plane PCB noise – Type a floating point value to specify the millivolt units of estimated board noise for bounce.
   - Power plane PCB noise – Type a floating point negative value to specify the estimated board noise for drop.
   - SSO Allowance – Specify an allowable percentage of SSO per bank.

   If any of the settings are estimated to be the same for all output signals, you can use the Allports row to quickly set the values for all outputs.
6. Choose **Design > Preference SSO Analysis**.

   The SSO calculator runs the analysis, based on the parameters, and displays the results in Package View.

   Any pins that failed the analysis are highlighted in red.

To view the full I/O SSO Analysis Report, select the Reports tab.

If some of the pin or bank assignments failed the analysis, adjust the assignments or parameters to try to reduce the SSO noise. Afterwards, rerun the SSO analysis.

See Also
- "Viewing SSO Analysis Results" on page 562
- "Analyzing SSO" on page 559
- "Exporting an SSO Report in CSV Format" on page 563
- "Running I/O SSO Analysis from the Command Line" on page 2578

Viewing SSO Analysis Results

After running SSO analysis, you can immediately open the I/O SSO Analysis Report, or you can export the results to a spreadsheet format file. You can also view the status of SSO pins in Package View. You do not have to save the SSO preferences in order to view or export the analysis.

To view the SSO Analysis Report:

- In the Report window, expand the Tool Reports folder and select **I/O SSO Analysis**.

The SSO Analysis Report is organized by "Bank SSO Results" and "Pin SSO Results." Each of these sections is further organized according to the type of change in current—ground bounce or drop—that is caused by the switching aggressor pins.

Bounce – When the aggressor pins switch from high to low, the power supply needs to sink current. If a large amount of current needs to be sunk, the ground level will start to rise. This rise in current is the ground bounce.

Drop – When the aggressor pins switch from low to high, the power supply needs to source current. If too much current is required, the VCC level will start to drop temporarily as the charge is supplied. This initial drop is the VCC drop. As the charge is supplied, the VCC level starts to go back up to the proper VCC level.

The “Pin SSO Results” section also contains hyperlinks to details organized by pin and bank.

To navigate to bank and pin details in the SSO Analysis Report:

- Click the desired bank number in the “Worst Bounce SSO of each Bank” section or the “Worst Drop SSO of each Bank” section to jump to the appropriate table in “SSO Information of Pins of each Bank”.
- Click a signal name in one of the tables of “SSO Information of Pins of each Bank” to jump to the detailed SSO Report by pin.

To view the SSO information for each output pin in Package View:

1. In Package View, choose **View > Show SSO View** or click the **SSO** button on the toolbar.

   On the pin layout, the SSO pins are highlighted with the status color. Those that failed the analysis are highlighted in red.

2. Hold your mouse pointer over an SSO pin to view the tool tip.

   The tool tip for each pin includes information from the SSO Analysis Report.

See Also
- “Analyzing SSO” on page 559
- “Running SSO Analysis” on page 561
- “Running I/O SSO Analysis from the Command Line” on page 2578
- “Exporting an SSO Report in CSV Format” on page 563

Exporting an SSO Report in CSV Format

After placement and routing, you can use Package View to export an SSO report as a comma-separated value file (.csv).

To export an SSO report in CSV format:

1. In Package View, choose **File > Export > SSO Report CSV File**.
2. In the dialog box, browse to the desired directory for the report.
3. Type a name in the file name box and click **Save**.

The SSO report is saved as a .csv file in the location you specified. You can view the report using a spreadsheet program that supports the CSV file format.

See Also ▶"Running SSO Analysis" on page 561

Exporting/Importing Pin Files and Preference Sheets

Diamond's Spreadsheet View supports delimiter-separated text formats, such as comma-separated value (CSV), to ease the viewing and editing of assignments and attributes. It also supports the exportation of a text version of all Spreadsheet View preferences. The following options are provided for exporting the contents of preference sheets:

Pin Layout File A pin layout file is a customizable report of pin information for your design. You can open the file in an external spreadsheet application for editing, and then import the edited pin layout file back into Spreadsheet View.

Lattice CSV File A Lattice CSV file is an exported preference sheet in CSV format. Preference sheets that can be exported include Port Assignments, Pin Assignments, Clock Resource, Route Priority, and Cell Mapping. You can edit the exported Lattice CSV file in an external spreadsheet program, and then import it back into Spreadsheet View.

Pinout File A pinout file is a comprehensive report, in CSV format, of all available and unbonded pins in the device.

LPF File of All Spreadsheet View Preferences This LPF file is a text file of all Spreadsheet View preferences, including default preferences.

See Also ▶AN8087, *Using Lattice Diamond Pin Layout Files and Pinout Files*

▶ "Exporting an SSO Report in CSV Format" on page 563

Exporting a Pin Layout File

A pin layout file is a customizable report of pin information for your design. Depending on the stage of the design flow, the pin layout file can be as simple as a list of available pins, pad names, functions, and banks. Or it might be a more detailed report that includes differential polarity, type, user assignments, default attributes, and custom column information. Finally, it can be a comprehensive report that also includes PAR assignments. You select the types of information you want included in the file, specify the order of presentation, and select the delimiter to be used as the value separator.

You can obtain a pin layout file from Spreadsheet View at any stage of the design flow. When you later import a pin layout file back into Spreadsheet View, all the information that was saved in the file is displayed.

To export a pin layout file:

1. In Spreadsheet View, save any assignments that are still in memory, and then choose **File > Export Pin Layout File**.

   The Export Pin Layout File dialog box opens and displays a list of all items that can be included in the report. Any custom columns that have been created are listed last, and you might need to scroll down to view them. You can also include pin migration information, if you have selected the Show Incompatible Pins option in Spreadsheet View or Package View. See "Migrating Pin Assignments" on page 484.

2. Select the items from the list that you want to appear as column headings in the file. Clear those that you do not want included.

   Note

   Pin_Number, Pad_Name, Function, and IO_Bank_Number cannot be cleared. These items will always be included in the pin layout file.

   If you select Pin Migration, a Migration column for each device that you selected in the Incompatible Pins dialog box will be included in the report. In each of these columns, a Y or N will indicate which pins are compatible or incompatible. Any Check Areas that were selected in the Incompatible Pins dialog box will not affect the report's contents.

3. Use the **Up** and **Down** buttons for selected items, as desired, to rearrange the order in which the column headings will appear in the report.

4. Select a value separator that will be used to separate the data into columns: comma, semicolon, space, or tab.

5. If you want the default values to be included for the user-assigned signals in the report, select the **Export Default Value** option. If you do not want the default values included, clear this option.

   Note

   If your design has already been placed and routed, the Export Default Value option will be dimmed.

6. Do one of the following:

   - To export the pin layout file to your current project directory, type a a name in the File Name box. If you are using the comma as the delimiter, include the .csv extension.
   - To export the pin layout file to a different directory, click **Browse**, navigate to the desired directory, and type a name in the File name box. If you are using a delimiter other than the comma, choose **All Files** from the "Save as type" drop-down menu; other wise, keep the default .csv extension. Click **Save**.

7. Click **OK**.

The pin layout file is saved to your project directory or to the directory you specified. When you open the pin layout file in a spreadsheet program, the columns will appear in the order you selected, including any custom columns. You can edit the cell contents of the columns as desired and import the file back into Spreadsheet View.

See Also ▶AN8087, *Using Lattice Diamond Pin Layout Files and Pinout Files*

- "Exporting a Pinout File in CSV Format" on page 568
- "Importing a Pin Layout File" on page 566
- "Exporting and Importing Preference Sheets" on page 567
- "Migrating Pin Assignments" on page 484
- "Adding Custom Columns" on page 420

Importing a Pin Layout File

A pin layout file can be imported into Spreadsheet View after the Translate Design process.

To import a pin layout file:

1. In Spreadsheet View, choose **File > Import Pin Layout File**.
2. In the dialog box, select the same value separator that was used in the pin layout file when it was exported.
3. Click the **Browse** button to open the dialog box and navigate to the pin layout file. If the file is not a comma-separated-value (.csv) file, select **All Files** from the "Files of type" drop-down menu.
4. Select the file and click **Open**.
5. Click **OK**.

   If the pin layout file contains custom columns, the Import Custom Columns dialog box will appear. If you do not want to import custom columns, click **Skip**.

   To import custom columns, do the following:

   a. Select the custom columns that you want to import.

   b. For each custom column, select Port Tab or Pin Tab from the Dest. Tab list.

   c. If desired, change the name of one or more of the selected custom columns.

   d. Click **OK**.

Spreadsheet View displays the pin assignments and attributes from the imported pin layout file.

Note

If you selected a value separator that is not in the pin layout file, you will receive an error message that the file cannot be imported. If this happens, verify the correct value separator and begin the procedure again.

See Also ▶"Exporting a Pin Layout File" on page 564

- "Exporting a Pinout File in CSV Format" on page 568
- AN8087, *Using Lattice Diamond Pin Layout Files and Pinout Files*

Exporting and Importing Preference Sheets

Diamond enables you to export the contents of an active preference sheet as a Lattice comma-separated value (CSV) file, edit the preferences in a preferred spreadsheet program, and then import the edited file back into Spreadsheet View. Preference sheets that can be exported in CSV format include Port Assignments, Pin Assignments, Clock Resource, Route Priority, and Cell Mapping.

To export a preference sheet as a Lattice CSV file:

1. In Spreadsheet View, make sure that the preference sheet you want to export is active. If not, click the tab at the bottom of the Preference Sheet pane to display the desired sheet.
2. Choose **File > Export > Lattice CSV File**.
3. In the "Export Sheet in CSV format" dialog box, specify the file name and desired location, and then click **Save**.

   The contents in the active preference sheet is saved as a .csv file with the name and location you specified. You can edit it with another spreadsheet program that supports the CSV file format.

   Note

   When you export a Lattice CSV file of the Port Assignments sheet, the CSV file will contain an extra column, "I/O Direction." This column will not be displayed when you import the CSV file back into Spreadsheet View.

After you have exported a preference sheet as a comma-separated value file (.csv), you can import it back into Spreadsheet View and edit the contents.

To import a Lattice CSV preference sheet:

1. In Spreadsheet View, choose **File > Import > Lattice CSV File**.
2. In the "Import Sheet in CSV format" dialog box, browse for the existing .csv file that you want to import and click **Open**.

The imported file is displayed, replacing the original contents of the related preference sheet.

See Also ▶"Exporting a Pin Layout File" on page 564

▶ "Exporting a Pinout File in CSV Format" on page 568

Exporting a Pinout File in CSV Format

The pinout file is a report of all device pins in the targeted device. It includes information such as pin/ball function, pin number, type, bank, dual function usage, and differential type. Information is included for all available pins and unbonded pins. The file is exported in comma-separated value (.csv) format, which enables you to examine it in an external spreadsheet application. You can export a pinout file from Spreadsheet View at any stage of the design flow.

To export a pinout file:

1. In Spreadsheet View, choose **File > Export > Pinout File**.
2. In the "Export Pinout File" dialog box, specify the file name and location that you want, and then click **Save**.

   The information is saved as a .csv file with the name and location you specified. You can view the file using a spreadsheet program that supports the CSV file format.

   Note

   The pinout file cannot be imported into Spreadsheet View.

See Also ▶"Exporting a Pin Layout File" on page 564

▶ AN8087, *Using Lattice Diamond Pin Layout Files and Pinout Files*

Exporting All Spreadsheet View Preferences

Diamond enables you to export all Spreadsheet View preferences to a single .lpf file. The new .lpf file will include all preferences that are still in memory, those from the HDL, and those that you have saved to the currently active .lpf file. It will also include default values for all preferences that you have not set for the design. Unlike the exportation of a pin layout file, exporting all Spreadsheet View preferences does not require that you save in-memory preferences before you export.

This is a quick way to save all of the preference settings so that they can be easily be modified and used in the current design or reused in another design. To use the new .lpf file, add it to your project and activate it for an implementation.

To export all Spreadsheet View preferences to an .lpf file:

1. In Spreadsheet View choose **File > Export > All SSV Preferences to LPF**.
2. In the dialog box, type a file name for the .lpf file.

   The file name must be different than the name of the active .lpf file in your project. If you try to save it using the same name as the active .lpf file, you will get an error message, even if you use a different directory.
3. Click **Save**.

   Diamond gathers all user assignments, plus default values for preferences that have not been set, and places them in the new file.

See Also ▶"Managing Constraint Files" on page 41

▶ "Exporting and Importing Preference Sheets" on page 567

Adding FPGA Attributes to HDL

HDL attributes that you specify in the source code are passed through logic synthesis into the EDIF netlist. The Translate Design process passes them to the native generic database (.ngd), and they are finally extracted or interpreted by the Map Design process (design mapper). Most attributes are extracted by the mapper and are written into the SCHEMATIC START/END section of the project's physical preference file (.prf). These are then used by PAR to update the native circuit description (.ncd). Some attributes, such as IO_TYPE, are interpreted directly by the mapper and are applied as part of the physical design file (.ncd) output.

To pass FPGA attributes to your EDIF netlist, use a synthesis tool. When edif2ngd reads the EDIF file, the attributes become part of the FPGA database.

The examples that follow show how the HDL code appears with an attached attribute and how the FPGA attributes appear in an EDIF netlist.

Note

Editing the EDIF file directly is not recommended in most cases, because the attributes should be present in the original HDL design in order to maintain design integrity. If you edit the EDIF file, it will no longer match the HDL. Consequently, design adjustments that you make later in the design flow will be compromised. When you re-synthesize your design, the edited EDIF files are overwritten. For test purposes, you might want to edit the EDIF file and run it through the flow. An example is provided here that shows where attributes occur in the resulting EDIF output.

Figure 104: Example 1 - Verilog Using Synplify

```
IB I_0 (.I(I[0]), .O(X[0])) /* synthesis IO_TYPE="LVDS" */ /*
synthesis PULLMODE="NONE" */ ;
```

The input buffer (IB) element is instantiated with an IO_TYPE attribute with a value of “LVDS,” which specifies the buffer type. The attributes are specified at the instance line before the semicolon.

Figure 105: Example 2 - Verilog Using Precision

```
  IB I_0 (.I(I[0]), .O(X[0]));
  ...

  // pragma begin
  // pragma attribute IO_TYPE LVDS
  // pragma attribute I_0 PULLMODE NONE
  // pragma end
endmodule
```

The attributes for all the instantiations are specified at the end just before endmodule.

Figure 106: Example 3 - VHDL Output

```
attribute IO_TYPE : string;
attribute PULLMODE : string;
attribute IO_TYPE of I_0 : label is "LVDS";
attribute PULLMODE of I_0 : label is "NONE";
...
I_0 : IB
  port map (I => I(0),
            O => X(0));
```

The attributes are defined and specified in the architecture declaration section. The VHDL syntax is the same for Synplify and Precision.

Figure 107: Example 4 - EDIF Netlist

```
(instance IZ0Z_0 (viewRef verilog (cellRef IB))
(property IO_TYPE (string "LVDS"))
(property PULLMODE (string "NONE"))
```

The input buffer (IB) element in Example 2 shows how it would appear in an EDIF netlist. Notice that the attribute appears enclosed in parentheses in the following format below the instance line:

(property <*attribute*> (string “<*parameter*>”))

See Also ▶“FPGA Libraries Reference Guide” on page 1665

- Synplify Pro for Lattice User Guide
- Synplify Pro for Lattice Reference Manual

Achieving Timing Closure

Setting meaningful constraints is one of the most important strategies for meeting timing goals. Attributes and preferences work together with synthesis, mapping, placement and routing to favorably impact final design timing.

The following steps address the tools and related information available for each stage of the design flow.

1. Generate meaningful and efficient HDL code for the specific architecture to guide the synthesis tools. Though many popular synthesis tools have significantly improved FPGA optimization algorithms, front-end design entry coding still plays an important role in utilizing FPGA resources.
2. Use synthesis tool options to improve the EDIF file. In some designs, this is essential for meeting the desired target.
3. Use FPGA preferences to specify the design timing goals that you want targeted during the Place & Route Design process. Along with a good functional design, a good set of timing preferences is crucial for meeting timing goals and for proper device operation on the system board. Preferences can be entered and modified at multiple points of the Diamond process flow.
4. Leverage Place & Route Design process properties, such as increased placement effort level and multiple routing passes. While the default settings address most designs in the most time-efficient manner, timing-critical designs may require the extra effort that such options provide.
5. If you cannot meet the performance goals of your design with timing preferences and additional effort levels of the Place & Route Design process, you can still improve performance by directing the physical layout of the circuit in the FPGA. This step, often referred to as floorplanning, is done by specifying FPGA location preferences.
6. Use location preferences to keep certain key logic together (UGROUP/HGROUP) and, optionally, to keep them in a particular area of the device (REGION/LOCATE). This is especially useful in creating an effective data flow direction in a datapath-intensive design. Your goal is to create an optimized placement starting point for the timing-critical portion of the design. The remaining logic will be placed by the tools to meet timing goals.
7. Use Diamond's preference views and integrated timing analysis to find the root cause of long timing paths. After problem areas are identified, your goal is to guide placement by refining LOCATE, REGION, and UGROUP preferences or by developing additional ones.
8. Use the ECO Editor, to customize your design implementation. The ECO Editor provides intimate access to routing control and the PFU/PFF array of the chip design. At this point a Lattice factory representative is usually involved to help you meet your objectives.

Chapter 7

Implementing the Design

Design implementation includes the processes of synthesizing and translating the design to build the internal database, mapping the design to device-specific components, assigning the mapped components to specific locations, and establishing physical connections that will join the components in an electrical network. The Diamond environment enables you to set options for each of these processes to help optimize results. Such options might include mapping, map register retiming, placement effort level, and placement and routing, among others. These options are available in the active strategy for the design. For descriptions of all design implementation process options, see the "Strategy Reference Guide" on page 1112.

Defining a device pinout is also an important part of design implementation. The I/O Assistant strategy enables you to produce a valid device pinout, based on board layout requirements, before the design is complete. The output of the I/O Assistant flow can then be back annotated to the logical preference file. See "Validating an I/O Plan" on page 573 for more information about the I/O Assistant flow. When you run an implementation process, the Diamond software automatically generates a report of the results and makes it available in the Reports window. Diamond will also generate timing reports, including Map Trace, Place & Route Trace, and I/O Timing Analysis when you have selected these options in the Process view.

See Also

- "Using Strategies" on page 48
- "Synthesizing the Design" on page 578
- "Translating the Design Database" on page 588
- "Mapping" on page 590
- "Place and Route" on page 602
- "Applying Design Constraints" on page 360
- "Command Line Reference Guide" on page 2464

Validating an I/O Plan

Defining a device pinout can be a complicated process because of constraints in the PC board layout and the FPGA architecture, and it is typically done long before the entire FPGA design is complete. The I/O Assistant, a special predefined strategy within Diamond, assists you with this task, enabling you to produce an FPGA-verified pinout early on based upon PC board layout requirements.

The I/O Assistant strategy helps you select a legal device pinout and produce LOCATE and IOBUF preferences for optimal I/O placement. It helps you accomplish this without completing the entire design. In fact, the only design content required to validate an I/O plan is an HDL model of the I/O ports. Details of the internal logic can be treated as a black box. The primary output of the I/O Assistant flow is a validated placement of I/O signals that can be back annotated to the logical preference file.

I/O Assistant Design Flow

The following sequential steps are typical for the I/O Assistant design flow:

1. Create a top-level module in HDL that describes all of the ports in the design. You can do so manually or use the I/O modules generated by IPexpress.
2. In Diamon, make the I/O Assistant strategy the active one for your project. From the Strategies folder in the File list pane, right-click **I/O Assistant** and choose **Set as Active Strategy**.

   The I/O Assistant strategy is a read-only predefined set of properties for the design flow.

   Note

   The I/O Assistant flow does not include exporting a JEDEC or bistream file. If you attempt to use the Export JEDEC File or Export Bitstream File process while using the I/O Assistant strategy, the process will fail.

3. Synthesize your HDL as you would normally.
   - If you are using Synplify Pro, Diamond will automatically pass the required attributes and header files for I/O Assistant flow when you run theTranslate Design process.
   - If you are running synthesis in stand-alone mode, you will need to include these attributes and header library files in the source code before synthesis.
4. Constrain your design to add banking location preferences, I/O types, I/O ordering, and minor customizations. You can set these preferences using Diamond's Spreadsheet View or you can do this manually.
   - To set the preferences in Spreadsheet View, choose **Tools > Spreadsheet View** and edit the I/Os.

     See "Assigning Signals" on page 466 and "Setting Port, Net, and Cell Attributes" on page 475.

- To set I/O preferences manually, double-click the name of the project's logical preference file (.lpf) from the LPF Constraint Files folder in the File List pane.

5. Run the **Place & Route Design** process.

   The process maps and places the I/Os based on the preferences, the I/O Assistant strategy, and the architectural resources. The output is a pad report (.pad) to guide future placement and a placed and routed native circuit description (.ncd) that contains only I/Os.

6. Examine the I/O Placement results by doing one or more of the following:
   - From the Process Reports folder in the Reports view, select **Signal/ Pad** to open the PAD Specification File and examine the pinout.
   - Choose **Tools > Package View**, and then choose **View > Display IO Placement** to view the pin assignments on the layout to cite areas for minor customization.
   - To view the results of timing constraints:
     - Run the **Place & Route Trace** process and open the Place & Route Trace report from the Reports window.
     - Run the **I/O Timing Analysis** process and open the I/O Timing Report from the Reports window.
7. Make any needed adjustments to the I/O preferences, as you did in Step 4.
8. Rerun **Place & Route Design**.
9. Repeat steps 5 through 7as necessary to achieve your I/O placement objectives.
10. From Package View, Spreadsheet View, or Floorplan View, choose **Design > Back Annotate Assignments** to copy the I/O preferences to the logical preference file, and then choose **File > Save**.

    I/O placement preferences are written to the end of the .lpf file and will take precedence over any existing preferences that may conflict with them.

11. In Diamond, create a new strategy or add an existing one. Set the strategy as the active one, and take your design through the regular flow.

See Also "I/O Assistant Design Entry" on page 574

I/O Assistant Design Entry

The I/O Assistant flow begins with an HDL model of the I/O plan. Create your HDL template files and assemble your HDL into a full chip-level HDL design. IPexpress can provide HDL templates of specialty interfaces required for DDR memories. In IPexpress, produce a module HDL template, perform instantiation of any necessary clocking structure, and any further customization.

The HDL design does not need to be complete. Input nets can be left hanging. The design tools will not optimize away dangling nets or outputs that are not driven.

The HDL model may contain I/Os only. Synthesis directives are applied to force I/O buffers to be inferred and prevent the incomplete model from being optimized away.

syn_force_pads Attribute If you are using Synplify Pro in the Diamond environment or in stand-alone mode, you must include the syn_force_pads attribute in the top level of your HDL source. This attribute prevents Synplify from optimizing away user-instantiated pads. It also causes Synplify to insert pads on unconnected ports, insert bidirectional pads on ports declared as inouts, and insert output pads on unconnected outputs as opposed to tristate pads.

If you are running Synplify in stand-alone mode, you will also need to include the following attributes and header library files in the source code.

syn_noprune Attribute This attribute ensures that the unconnected I/O buffers in a partial design are not optimized away.

The following Synplify code examples illustrate the synthesis attributes that enable the I/O Assistant flow to run a partial design through the Diamond software. The module cannot be empty. At least some logic or dummy equation must be present; otherwise, Synplify interprets the architecture as a black box model.

Figure 108: Verilog Example

```
module ioa1(rxclk, rxd,
            txclk, txd) /* synthesis syn_force_pads=1
syn_noprune=1*/;
input rxclk;
input  [15:0]  rxd;
output  txclk;
output [15:0]  txd;
wire xy;
assign xy = 1'b0;  //dummy equation
endmodule
```

Running I/O Assistant Flow from the Command Line

To run your design through an I/O Assistant design flow using the command line, proceed initially as you would for a regular command-line flow. In Map and PAR, however, use the special switches described in Steps 5 and 6 that will allow the pinout to be selected early in the design process without having a complete design.

1. Complete the top-level design module and synthesize the design.

   See “I/O Assistant Design Entry” on page 574.

Figure 109: VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
entity ioa1 is port (
  rxclk : in std_logic;
  rxd : in std_logic_vector(15 downto 0);
  txclk : out std_logic;
  txd : out std_logic_vector(15 downto 0)
);

end ioa1;
architecture rtl of ioa1 is
attribute syn_force_pads : boolean;
attribute syn_force_pads of rtl : architecture is true;
attribute syn_noprune : boolean;
attribute syn_noprune of rtl : architecture is true;
signal dummy : std_logic;
begin
dummy <= rxd(0);
end rtl;
```

2. Run **edif2ngd** on your top-level EDIF file from the command line. This command outputs an .ngo file for input into **ngdbuild**. For example:

   ```
   edif2ngd -l or5s00 toplevel.edn
   ```

3. Run **ngdbuild** on your top-level native generic database (.ngd) file from the command line. This process uses the input .ngo file to output the .ngd file. For example:

   ```
   ngdbuild -a or5s00 toplevel.ngo
   ```

   Syntax:

   edif2ngd -l <*library_name*> <*source*>.edn

   ngdbuild -a <*arch_name*> <*input*>.ngo -mc

   where:

   -l switch indicates the library for a specific architecture.

   -a switch indicates the architecture name follows.

4. Set preferences for the types of I/Os and relative locations on the device package, using the Diamond's Spreadsheet View or a text editor, and save them to the logical preference file (.lpf).

   See "Assigning Signals" on page 466 and "Setting Port, Net, and Cell Attributes" on page 475.

5. Run **map** on your .ngd and .lpf files, using the -u switch, to produce a physical native circuit description (.ncd) file.

   The –u switch indicates to the map program to leave unconnected logic in the design. This allows a design to consist solely of the I/Os without

internal logic. This allows the pinout to be selected early in the design process without having a complete design. For example:

```
map -a or5s00 -p or5s25 -u toplevel.ngd -o toplevel.ncd
toplevel.lpf
```

The map program creates a physical database, .ncd, which is passed onto the par program to assign the pinout.

6. Run **par** on your physical design (.ncd) file from the command line, using the -io switch.

   The –io switch indicates to the par program to only route connections to the I/O logic and I/O buffers in the design. This is mainly to save run time of par. It might be necessary in some cases to not use the –io switch and let par route the entire design when selecting a pinout. This depends on how complete the design is and if the pinout needs to be verified when working in a complete design environment. For example:

   ```
   par -io toplevel.ncd toplevel_par.ncd toplevel.prf
   ```

   The par program creates a .pad file that contains the results of the I/O pinout.

7. Review the .pad file to see if the I/O pinout is suitable for the board design.
8. Repeat steps 4 through 7until you have a suitable pinout and are satisfied that the design is ready for final placement and routing.
9. Copy the preferences from the end of the .pad file and paste them into the logical preference file.
10. Run the design through final placement and routing, using the regular command-line flow.

Figure 110: Sample Command Line DOS Batch File

```
set design=top
set edif=top.edn
edif2ngd -l LatticeSC %edif% %design%
ngdbuild -a LatticeSC %design%.ngo %design%.ngd
map -a LatticeSC -p LFSC25E -u %design%.ngd -o %design%_map.ncd
%design%.prf -pr %design%_map.prf
par -w -io %design%_map.ncd %design%_par.ncd %design%_map.prf
```

See Also
- “Running EDIF2NGD from the Command Line” on page 2487
- “Running NGDBUILD from the Command Line” on page 2492
- “Running MAP from the Command Line” on page 2495
- “Running PAR from the Command Line” on page 2505

Synthesizing the Design

Synthesis is the process of translating a register-transfer-level design into a process-specific, gate-level netlist that is optimized for Lattice Semiconductor FPGAs.

In different ways, Diamond can be used with almost any synthesis tool. Diamond comes with two tools fully integrated: Synopsys Synplify Pro for Lattice and Lattice Synthesis Engine (LSE). “Fully integrated” means that you can set options and run synthesis entirely from within Diamond. Diamond also supports Mentor Graphics Precision RTL Synthesis but does not include the tool itself. If you want to use Precision RTL Synthesis, you need to get the software on your own. If you prefer, you can use other synthesis tools by running them independently of Diamond.

Synthesis tools can be run three ways with Diamond:

- Integrated synthesis is the simplest way. You work entirely within Diamond. See “Integrated Synthesis” on page 583.
- Interactive synthesis provides much greater control of how synthesis is done. You set up a design project in Diamond, but set up and run synthesis directly in the synthesis tool. See “Interactive Synthesis” on page 585.
- Stand-alone synthesis allows use of other, non-supported synthesis tools. You set up and run synthesis directly in the synthesis tool and then Diamond imports the result. See “Stand-Alone Synthesis” on page 586.

LSE can only be run in integrated synthesis.

Whichever method you use, see “Pre-Synthesis Check List” on page 578 to make sure your project is ready to synthesize.

See Also

- “Selecting a Synthesis Tool” on page 89
- TN1008, “HDL Synthesis Guidelines for Lattice Semiconductor FPGAs” for how coding style influences performance and area utilization
- Synplify and Synplify Pro for Lattice User Guide
- Synplify and Synplify Pro for Lattice Reference Manual

Pre-Synthesis Check List

Following is a list of tasks to be done before running synthesis. Most of these are normally done as parts of other tasks, such as setting up the project and design entry.

In all cases, in the source code, add or adjust the constraints, attributes, and directives used by the synthesis tool. See the synthesis tool’s user documentation. For LSE, see “Lattice Synthesis Engine (LSE) Constraints” on page 1339.

If using PMI modules:

- Add the PMI synthesis header file, pmi_def.v or pmi_def.vhd, to the project. See “PMI Synthesis Header Files” on page 202.
- If using PMI modules in VHDL code, compile the PMI component declaration file into your work library. See “VHDL Requirements” on page 202.

Integrated synthesis:

- Specify a synthesis tool. See “Selecting a Synthesis Tool” on page 89.
- Specify the top-level unit. See “Setting the Top-Level Unit for Your Project” on page 43. This is not always required but is a good practice. If not specified, you are relying on the defaults of the synthesis tool.

  Note

  In VHDL and mixed-language designs, LSE stops with an error if the top-level unit is not specified. LSE can find the top-level unit in pure Verilog designs.

- Specify a search path for files referenced by Verilog include directives. See “Specifying Search Path for Verilog Include Files” on page 90.
- Specify a VHDL library name. The default is “work.” See “Specifying VHDL Library Name” on page 90.
- Order the source files for synthesis. The synthesis tool processes the files in the order showing in the File List frame.
- Specify the strategy settings for the synthesis tool. See “Using Strategies” on page 48. If using LSE, also see “Optimizing LSE for Area and Speed” on page 581.

Interactive and stand-alone synthesis:

- Reference the Lattice synthesis header library in the source code. See “Lattice Synthesis Header Libraries” on page 579.
- If using guided mapping or guided place-and-route, set the synthesis tool to minimize changes to signal names, such as with an incremental flow.
- For interactive synthesis, if you want to use the Synplify Pro strategy settings instead of Synplify Pro’s own defaults, change the “Export Diamond Settings to Synplify Pro GUI” option to **Yes** or **Only on First Launch**. See “Export Diamond Settings to Synplify Pro GUI” on page 1120.

Lattice Synthesis Header Libraries

The synthesis header libraries define primitives from the FPGA Libraries. A separate library is available for each device family and in Verilog and VHDL versions. The header libraries support both Precision and Synplify synthesis tools. If the design has any primitives from the FPGA Libraries instantiated, the appropriate header library is required.

The integrated flow automatically includes the FPGA libraries, but if you are running synthesis outside of Diamond with interactive or stand-alone synthesis, you need to add a reference to the appropriate library manually. Look in Table 46 for the device family that you are using and add it to the source file list of your synthesis project. If your design is VHDL-based, add the .vhd file. If your design if Verilog-based, add the .v file.

The files are located at:

<*install_dir*>/cae_library/synthesis/verilog
<*install_dir*>/cae_library/synthesis/vhdl

If your design is in VHDL, also specify the library name for the header file.

Table 46: Synthesis Header Files

Device Family	Header Library File	Library Name
ECP5U	ec5u.v eco5u.vhd	ecp5u
ECP5UM	ec5um.v eco5um.vhd	ecp5um
LatticeEC	ec.v ec.vhd	ec
LatticeECP	ecp.v ecp.vhd	ecp
LatticeECP2	ecp2.v ecp2.vhd	ecp2
LatticeECP2S	ecp2s.v ecp2s.vhd	ecp2s
LatticeECP2M	ecp2m.v ecp2m.vhd	ecp2m
LatticeECP2MS	ecp2ms.v ecp2ms.vhd	ecp2ms
LatticeECP3	ecp3.v ecp3.vhd	ecp3
LatticeSC	sc.v sc.vhd	sc
LatticeSCM	scm.v scm.vhd	scm
LatticeXP	xp.v xp.vhd	xp
LatticeXP2	xp2.v xp2.vhd	xp2
LIFMD	lifmd.v lifmd.vhd	lifmd

Table 46: Synthesis Header Files (Continued)

Device Family	Header Library File	Library Name
LIFMDF	lifmdf.v lifmdf.vhd	lifmdf
MachXO	machxo.v machxo.vhd	machxo
MachXO2	machxo2.v machxo2.vhd	machxo2
MachXO3D	machxo3d.v machxo3d.vhd	machxo2d
MachXO3L	machxo3l.v machxo3l.vhd	machxo3l
Platform Manager	lptm.v lptm.vhd	lptm
Platform Manager 2	lptm2.v lptm2.vhd	lptm2

Optimizing LSE for Area and Speed

The following strategy settings for Lattice Synthesis Engine (LSE) can help reduce the amount of FPGA resources that your design requires or increase the speed with which it runs. (For other synthesis tools, see those tools' documentation.) Use these methods along with other, generic coding methods to optimize your design. Also, consider using the predefined Area or Timing strategies.

Minimizing area often produces larger delays, making it more difficult to meet timing requirements. Maximizing frequency often produces larger designs, making it more difficult to meet area requirements. Either goal, pushed to an extreme, may cause the place and route process to run longer or not complete routing.

To control the global performance of LSE, modify the strategy settings. Choose **Project > Active Strategy > LSE Settings**. In the Strategy dialog box, set the following options, which are found in Synthesize Design > LSE. See the following text for explanations and more details.

Table 47: LSE Strategy Settings for Area and Speed

Option	Area	Speed
FSM Encoding Style	Binary or Gray	One-Hot
Max Fanout Limit	<*maximum*>	<*minimum*>
Optimization Goal	Area	Timing
Remove Duplicate Registers	True	False
Resource Sharing	True	False

Table 47: LSE Strategy Settings for Area and Speed

Option	Area	Speed
Target Frequency	*<minimum>*	
Use IO Registers	Auto or True	Auto or False

FSM Encoding Style If your design includes large finite state machines, the Binary or Gray style may use fewer resources than One-Hot. Which one is best depends on the design. One-Hot is usually the fastest style. However, if the finite state machine is followed by a large output decoder, the Gray style may be faster.

Max Fanout Limit A larger fanout limit means less duplicated logic and fewer buffers. A lower fanout limit may reduce delays. The default is 1000, which is essentially unlimited fanout. Select a balanced fanout constraint. A large constraint creates nets with large fanouts, and a low fanout constraint results in replicated logic. You can use this in conjunction with the syn_replicate attribute. See “syn_replicate” on page 1383. To minimize area, don’t lower this value any more than needed to meet other requirements. To minimize speed, try much lower values, such as 50.

You can change the fanout limit for portions of the design by using the syn_maxfan attribute. See “syn_maxfan” on page 1372. Set Max Fanout Limit to meet your most demanding requirement. Then add syn_maxfan to help other requirements.

Optimization Goal If set to Area, LSE will choose smaller design forms over faster whenever possible.

If set to Timing, LSE will choose faster design forms over smaller whenever possible. If a create_clock constraint is available in an .ldc file, LSE ignores the Target Frequency setting and uses the value from the create_clock constraint instead.

If you are having trouble meeting one requirement (area or speed) while optimizing for the other, try setting this option to **Balanced**.

Note

The minimum pulse width associated with devices is not checked by the synthesis timing engine. The place and route timing engine checks if a given clock definition is within the minimum pulse width requirement of the devices that the clock drives.

Remove Duplicate Registers Removing duplicate registers reduces area, but keeping duplicate registers may reduce delays.

Resource Sharing If set to True, LSE will share arithmetic components such as adders, multipliers, and counters whenever possible.

If the critical path includes such resources, turning this option off may reduce delays. However, it may also increase delays elsewhere, possibly reducing the overall frequency.

Target Frequency A lower frequency target means LSE can focus more on area. A higher frequency target may force LSE to increase area. Try setting this value to about 10% higher than your minimum requirement. If Optimization Goal is set to Timing and a create_clock constraint is available in an .ldc file, LSE will use the value from the create_clock constraint instead.

Use IO Registers If set to True, LSE will pack all input and output registers into I/O pad cells. Register packing reduces area but adds delays.

Auto, the default setting, enables this register packing if Optimization Goal is set to Area. If Optimization Goal is Timing or Balanced, Auto disables register packing.

You can also control packing on individual registers. See "syn_useioff" on page 1398. Set Use IO Registers to meet your most demanding requirement. Then add syn_useioff to help other requirements.

See Also

- "Using Strategies" on page 48
- "LSE Options" on page 1129

Integrated Synthesis

In integrated synthesis, you create, synthesize, and implement a design completely within the Diamond environment. This is the simplest method, but it limits your control of synthesis to tools and features directly supported by Diamond.

Note for Lattice Synthesis Engine (LSE)

LSE does not produce the usual EDIF file. Instead, LSE takes the next step, translating the design into an .ngd file ready for mapping.

Also, LSE produces a Verilog netlist suitable for simulation. The file is placed in the design implementation's folder and is named <*top_module*>_prim.v.

To synthesize your design in Diamond:

If you haven't already, check the Pre-Synthesis Check List before running synthesis.

1. If desired, change the strategy settings. Choose **Project > Active Strategy >** <*Synthesis Tool*> **Settings**. See also "Using Strategies" on page 48.

   Note

   If you are switching from interactive synthesis to integrated, you need to reset all the options in Diamond. Options set directly in the synthesis tool have no effect in Diamond.

2. In the Process view, double-click **Synthesize Design** or any following stage. Double-clicking a following stage runs Synthesize Design, if necessary, and any other stages up to the one you clicked.

   Note for LSE

   If this is the first time running synthesis on the design or if you changed the .ldc file, double-click **Synthesize Design** only. You may need to adjust your .lpf file before running Map Design (see the final step below).

   If Synthesize Design has already been run and does not recognize any changes in the design, it won't run again. You can force Synthesize Design to run by right-clicking it and choosing **Rerun** from the pop-up menu.

3. When finished, check the icon next to Synthesize Design in the Process frame. A green check mark indicates success; a yellow triangle indicates success with warnings; a red X indicates failure.

4. For more information on how the synthesis ran, open the **Reports** view in the Tools area (choose **View > Reports**). Under Design Summary, choose **Process Reports >** <*Synthesis Tool*>. This report lists actions taken by the synthesis tool, warnings, and errors. The report also provides a list of FPGA resources used and a timing report based on estimated place-and-route data. See "Viewing Logs and Reports" on page 101.

   To quickly find warnings or errors, click the **Warning** or **Error** tab in the Logs area. The Output tab includes the same information that is in the detailed report.

   You can find a more detailed area report in the folder of the active implementation. The report is named with an.areasrr extension (for Synplify Pro) or with an .arearep extension (for LSE). The report includes the resources used by each module of the design. Similar information can also be found in the Hierarchy view.

   If you are using LSE, you can also find a more detailed timing report in the folder of the active implementation. The report is named <*top_module*>_lse.twr. It is similar to the report described in "TRACE Report File" on page 700.

5. If you are using LSE with an .ldc file and set Use LPF Created from SDC in Project to True in the strategy settings, the output includes an .lpf file that has preferences equivalent to those in the .ldc file. The .lpf file is placed in the design implementation's folder and is named <*top_module*>_lse.lpf. You can make these preferences available to later stages of the design implementation process by merging <*top_module*>_lse.lpf into your active .lpf file. See the following procedure.

To merge LSE's .lpf into the active .lpf:

1. Go to the File List view and open the folder for the active implementation. The folder uses this icon, , and the name is in bold text.

2. Under Constraint Files, double-click the active .lpf file.

   Source Editor opens with the .lpf file.

3. In the main window, choose **File > Open > File**.
4. In the Open File dialog box, in the “Files of type” drop-down menu at the bottom, choose **Preference Files**.
5. Browse to the implementation folder and select <*top_module*>**_lse.lpf**.
6. Click **Open**.

   Source Editor opens a second view with the .lpf file. This is the .lpf created by LSE. Besides the preferences based on the synthesis constraints, the .lpf file automatically includes two BLOCK preferences that are added automatically to .lpf files.
7. Choose **Window > Split Tab Group**.

   The tools area of the main window splits into two groups. You should have the active .lpf on one side and <*top_module*>_lse.lpf on the other side. If they are both on the same side, select the tab of one and drag it to the other side.
8. In <*top_module*>_lse.lpf, select everything from line 3 down. This is everything except the first two BLOCK preferences.
9. Press Ctrl-c to copy the selected preferences.
10. Go to the active .lpf and place the cursor at the bottom of the file, in a blank line beneath all the text.
11. Press Ctrl-v to paste in the new preferences.

    Note

    If the active .lpf file already has preferences from a previous run of LSE, paste the new preferences over the old version. Do not create multiple copies of the same preferences.
12. Save the active .lpf file and close both files.

See Also

- “Optimizing LSE for Area and Speed” on page 581
- “Running Processes” on page 70

Interactive Synthesis

In interactive synthesis, you set up a design project in Diamond, but set up and run synthesis directly in the synthesis tool. This gives you more complete control of synthesis than integrated synthesis and allows you to make full use of the synthesis tool’s features.

Note

If you decide to use interactive synthesis, remember to use it every time you make a design change. If you try to repeat the implementation process by simply double-clicking an item in the Process frame, Diamond will synthesize the design with integrated synthesis, which will probably give unexpected and inferior results.

LSE cannot be used in interactive synthesis.

To synthesize your design using interactive synthesis:

If you haven't already, check the "Pre-Synthesis Check List" on page 578 before running synthesis.

1. Open the synthesis tool from the **Tools** menu.

   The synthesis tool opens with the device and source files for your design.
2. Set options in the tool as desired.
3. If you choose to create a preference file, make sure it is not named <*project_name*>.lpf. That would overwrite the logical preference file created by Diamond.
4. To re-use the settings, save the synthesis tool's project file as <*project_name*>**_syn.prj** (Synplify) or <*project_name*>**.psp** (Precision) in the project folder.

   Make sure the Diamond strategy option, "Export Diamond Settings to Synplify Pro GUI" (under Synplify Pro in the Strategies dialog box) is set to **No** or **Only on First Launch**. See "Export Diamond Settings to Synplify Pro GUI" on page 1120.
5. Synthesize the design to create an EDIF file.
6. Analyze the results in the synthesis tool.
7. When you are satisfied with the synthesis, make sure the EDIF file is in your project directory and named <*project_name*>.edi. Synplify should do this automatically. With Precision you may need to move the file.

Stand-Alone Synthesis

In stand-alone synthesis, you can use a synthesis tool that is not directly supported by Diamond. In this flow you set up and run synthesis directly in the synthesis tool. Then create a design project in Diamond using the output of the synthesis tool as the source file.

Note

If you decide to use stand-alone synthesis, remember to use it every time you make a design change. If you try to repeat the implementation process by simply double-clicking an item in the Process frame, you probably will see nothing happen because the EDIF file has not changed.

LSE cannot be used in stand-alone synthesis.

If you haven't already, check the "Pre-Synthesis Check List" on page 578 before running synthesis.

To set up a project using a stand-alone tool:

1. Open the synthesis tool directly, without using Diamond.

2. Using your synthesis tool, create a project, including specifying a device, source files, and options.
3. Synthesize the design to create an EDIF file.
4. Analyze the results in the synthesis tool.
5. When you are satisfied with the synthesis, create a design project in Diamond.
6. Choose **File > Add > Existing File**.
7. In the Add Existing File dialog box, browse to the EDIF file and click **Open**.

To re-synthesize your design using the stand-alone tool:

This procedure assumes that you have already set up the project to use stand-alone synthesis.

1. Open the synthesis tool directly, without using Diamond.
2. In the synthesis tool, open the project.
3. If desired, change the options.
4. Synthesize the design to create an EDIF file.
5. Analyze the results in the synthesis tool.

Translating the Design Database

After you have created a project and imported the source files, the next step is to translate and build an internal database. The database is a logical network of device-independent components such as gates and flip-flops, and it is logically equivalent to your high-level source description. This step takes the hierarchy in the input netlist and translates and converts it into primitive terms that the Diamond system can understand and manipulate.

The input source file for this process is an EDIF file. The EDIF file can be generated from a stand-alone synthesis tool and imported into the Diamond environment as your design, or it can be created automatically from a Verilog or VHDL design using the integrated synthesis flow within the Diamond software.

When you run the translation and build processes (EDIF2NGD and NGDBUILD tools respectively), Diamond converts the EDIF file to a Native Generic Database (*.ngd) file. Both the translation and build processes are executed when you run the Translate Design process from the Process view in the Diamond environment.

If the design utilizes parameterized modules (PMI) or IP cores (Lattice modules and ispLeverCORE IP modules), or user firm macros, the cores are expanded in this process. The output .ngd file will be used later by the Map Design process to implement your design. The output of the translation and build processes is a logical Native Generic Database (.ngd) file.

About the Translate Design Process Given an HDL-based project file, the Translate Design process executes a logic synthesis phase using the synthesis tool you have configured to produce an EDIF file. It then performs the translations necessary on the EDIF to produce an .ngd file. If your project is EDIF-based, the build skips the logic synthesis phase and performs the EDIF-to-NGD translation only. The EDIF-to-NGD translation can be run from the command line using the edif2ngd and ngdbuild programs. For more information on HDL-based and EDIF-based projects, see “Managing Projects” on page 22.

The edif2ngd and ngdbuild processes of Translate Design perform netlist translation and netlist conversion. The netlist reader, edif2ngd, converts the netlist to an .ngo file. An .ngo file is a binary file that describes the design in terms of the components and hierarchy specified in the input design file. The ngdbuild program translates and merges one or more .ngo files, combining each component in the design to its equivalent Lattice FPGA primitives.

See Also

- “Running Translate Design” on page 590
- “Running EDIF2NGD from the Command Line” on page 2487
- “Running NGDBUILD from the Command Line” on page 2492

Setting Translate Design Options

When you run the Translate Design process in Diamond, the design is translated and the database built automatically, based on design options that are in the active strategy. You can use the Strategies dialog box to modify the Translate Design options before running a full or partial design flow.

To set Translate Design options in the Diamond environment:

1. Choose **Project > Active Strategy > Translate Design Settings** to open the Strategies dialog box.
2. For each option that you want to change, double-click a cell in the Value column. Select the desired value from the drop-down menu or click the browse button, as appropriate.
3. Click **OK**.

See Also ▶"Translate Design Options" on page 1138

- "Translate Design Input Files" on page 589
- "Translate Design Output Files" on page 590
- "Running Translate Design" on page 590

Translate Design Input Files

The following files are input to the Translate Design process:

- EDIF Netlist (.ed*) – Given an EDIF-based project file, the build reads the input netlist and expands any related PMI modules or macro files (.nmc) as part of the EDIF-to-NGD process.
- Verilog HDL Source File (.v) – Given a Verilog HDL-based project file, the build reads the input source files, performs logic synthesis to produce EDIF, then expands any related PMI modules or macro files (.nmc) as part of the EDIF-to-NGD process.
- VHDL HDL Source File (.vhd) – Given a VHDL-based project file, the build reads the input source files, performs logic synthesis to produce EDIF, then expands any related PMI modules or macro files (.nmc) as part of the EDIF-to-NGD process.

Note

For more information on PMI modules, see the "Lattice Module Reference Guide" on page 1400. For more information on .nmc physical macro files, see the Physical Macros topic in the EPIC Device Editor online Help.

See Also ▶"Translate Design Output Files" on page 590

Translate Design Output Files

The following files are created from the Translate Design process:

- Native Generic Database File (.ngd) – This is a generic database file that contains a logical description of the design in terms of Lattice library elements from structural models or inferred by logic synthesis. This converted file is the output of the ngdbuild process, which uses an .ngo input file that has been translated by a netlist reader (edif2ngd).
- Data Netlist Reader File (.ngo) – Output of the edif2ngd (netlist reader) subprocess, this data file contains a logical description of the design in terms of its original components and hierarchy. This translated file is the input to ngdbuild, which converts the file to describe it in terms of familiar system primitives.

See Also
- "Translating the Design Database" on page 588
- "Translate Design Input Files" on page 589
- "Running Translate Design" on page 590

Running Translate Design

In the Diamond environment, the Translate Design process builds the design database automatically.

To run Translate Design in the Diamond environment:

- In the Process view, double-click **Translate Design**.
- Alternatively, right-click **Translate Design** and choose **Run** from the pop-up menu.

To build your design database using the command line, use the edif2ngd and ngdbuild programs in a two-step process.

See Also
- "Setting Translate Design Options" on page 589
- "Running Processes" on page 70

Mapping

Mapping is the process of converting a design represented as a network of device-independent components, such as gates and flip-flops, into a network of device-specific components, such as PFUs and EBRs or configurable logic blocks.

You can use the Map Design process in the Diamond environment or the MAP program from the command line.

Using the logical Native Generic Database (.ngd) file created by the Translate Design process, the Map Design process generates physical descriptions of

the logical configuration within the programmable device elements. These device elements include programmable function units (PFU), programmable I/O cells (PIC), embedded block RAM (EBR). They also include special function blocks: internal oscillator, global set/reset (GSRN), start-up logic, phase locked loop (PLL), delay locked loop (DLL), and physical coding sublayer (PCS) logic.

Depending upon the attributes specified in the input netlist, mapping includes absolute placement, logical partitioning (hierarchical netlists), component group placement, and regional group placement information in the physical description. The resulting physical description is output to a physical native circuit description (.ncd) file, in terms of the components in the target architecture. This .ncd file can then be placed and routed using the Place & Route Design process.

Map and Preference Errors The Map process can be set up to stop whenever preference errors are encountered. This can be accomplished by setting the “Ignore Preference Errors” option to “False” in the Map Design section of the active strategy or by using the -pe option from the command line. This will cause the Map process to terminate and issue error messages when preference errors are found. By default, the Map process ignores preference errors.

Map and Trace In the Diamond Process view under Map Design, a Map Trace process is available that runs static timing analysis on the mapped .ncd file. This process reports any timing errors associated with preferences that have been set in the pre-map stage. You can view the Map Trace report in the Reports window by selecting it from the Analysis Reports folder. In addition, the processes Verilog Simulation File and VHDL Simulation File produce post-map source simulation files for functional simulation of primitive gate-level logic.

See Also ▶“Running MAP from the Command Line” on page 2495

- “Setting Map Design Options” on page 591
- “Mapping Input Files” on page 592
- “Mapping Output Files” on page 592

Setting Map Design Options

When you run the Map Design process in Diamond, the design is mapped based on design options that are in the active strategy. Mapping options can influence the performance and utilization of the design implementation, ease incremental design changes, cause the mapping process to stop when preference editors are encountered, or allow you to overmap in order to review how many resources are required for a particular implementation.

To set Map Design options:

1. Choose **Project > Active Strategy > Map Design Settings** to open the Strategies dialog.

2. For each option that you want to change, double-click a cell in the Value column. Select the desired value from the drop-down menu, enter text, or click the browse button, as appropriate.
3. Click **OK**.

See Also
- "Map Design Options" on page 1140
- "Mapping" on page 590
- "Mapping Input Files" on page 592
- "Mapping Output Files" on page 592

Mapping Input Files

The following files are input to the Map Design process.

- Native Generic Database File (.ngd) – This file is a logic description of the design in terms of Lattice library elements from structural models or inferred by logic synthesis produced by the Translate Design process.
- Logical Preference File (.lpf) – This file is the source file for storing constraints called logical preference. You can directly edit the .lpf file in Diamond's preference-editing views, the Source Editor, or an external text editor. When preferences are modified in the active .lpf file, the Map Design must be rerun.

See Also
- "Mapping" on page 590
- "Setting Map Design Options" on page 591
- "Mapping Output Files" on page 592
- "Using Diamond's Preference Views" on page 406
- "Setting Preferences" on page 459

Mapping Output Files

The following files are created from the Map Design process:

- Native Circuit Description File (.ncd) – The .ncd file is a physical description of the design in terms of the components in the target device.
- Physical Preference File (.prf) – The physical preference file is an ASCII output of the Map Design process. Preferences saved in the logical preference file (.lpf) are written to the .ncd file and to the .prf file during the map process. The map process rewrites the physical preference file each time it is executed. The .prf file is used as an input file to placement and routing and TRACE.
- MAP Report File (.mrp) — The map report file provides details about how the design was mapped to physical elements and reports any errors. Mapping details can include such things as how attributes were interpreted, logic that was removed or added, and how signals and

symbols in the logical design were mapped to components in the physical design.

See Also ▶"Mapping" on page 590

- "Setting Map Design Options" on page 591
- "Mapping Input Files" on page 592

Running Map Design

In the Diamond environment, the Map Design process automatically maps the design based on the active strategy settings and preferences in the active .lpf file.

To run Map Design in the Diamond environment:

- In the Process view, double-click **Map Design**.
- Alternatively, right-click Map Design and choose **Run** from the pop-up menu.

See Also ▶"Running MAP from the Command Line" on page 2495

- "Running Processes" on page 70

MAP Report File

The MAP Report (.mrp) file supplies statistics about component usage in the mapped design and shows the number and percentages of resources used out of the total resources in the device. The report is produced whether you have run Map in the Diamond environment or the command line.

To view the MAP Report file in Diamond:

- Click the **Reports** tab to activate the Report view and select **Map** in the Process Reports folder of the Design Summary pane.

  The Map report is displayed in the Reports window to the right.

  OR

- Using a text editor, open the .mrp ASCII file in your project directory.

Although detailed information varies depending on the device, the format of the .mrp file is the same. The report is divided into sections, which can include the following:

- Design Information – Shows the map command line, the LPF file path, the device/performance grade, and the time/date stamp of the run.

- Design Summary – Shows the number and percentage of resources used out of the total of the resources of the mapped device.

  Note

  The total resources reported represents the total resource count in the device and not the total accessible resources. Since the accessibility of resources can change, depending on the design and design revisions, there will sometimes be a gap between the number of resources that are available for use and the total number in the device.

  You will see a line showing the number of PFUs or Slices and whether they are implemented as logic/ROM/RAM or just logic/ROM. For older architectures that are PFU-based, you will see the number of PIO IDDR/ODDR, distributed RAM, ripple logic, shift registers, and many other items.

```
Design Summary
   Number of registers:    106 out of 13023 (1%)
      PFU registers:           85 out of 12420 (1%)
      PIO registers:           21 out of   603 (3%)
   Number of SLICEs:       125 out of  8280 (2%)
      SLICEs as Logic/ROM:    125 out of  8280 (2%)
      SLICEs as RAM:            0 out of  1485 (0%)
      SLICEs as Carry:          5 out of  8280 (0%)
   Number of LUT4s:        194 out of 16560 (1%)
      Number of logic LUTs:      184
      Number of distributed RAM:   0 (0 LUT4s)
      Number of ripple logic:      5 (10 LUT4s)
      Number of shift registers:   0
   Number of PIO sites used: 46 out of 201 (23%)
   Number of PIO FIXEDDELAY:    0
   Number of DQSDLLs:  0 out of 2 (0%)
   Number of PLLs:  2 out of 4 (50%)
   Number of block RAMs:  0 out of 15 (0%)
   Number of CLKDIVs:  0 out of 2 (0%)
   Number of GSRs:  1 out of 1 (100%)
   JTAG used :      No
   Readback used :  No
   Oscillator used :  No
   Startup used :   No
   Notes:-
      1. Total number of LUT4s = (Number of logic LUT4s) +
2*(Number of distributed RAMs) + 2*(Number of ripple logic)
      2. Number of logic LUT4s does not include count of
distributed RAM and ripple logic.
```

  The Number of registers in the sample report shows a total 106, derived from a total of 85 PFU registers and an additional 21 registers used in the IOLOGIC components in the Programmable I/O cell.

  The Number of SLICEs information in the Design Summary will vary in LatticeSC versus LatticeECP/EC and LatticeXP devices, which contain different types of slices. For LatticeECP2 and higher, Map has been enhanced to adjust its reporting to account for SLICEs of the PLC_Sr

variety, which are not usable for memory, so that the available resource capacity reporting is not misleading. A breakdown of LUT4 is given after this with a line that has the total number used in the design.

The number and percentage of PIO sites used, out of the total number, is given in the Design Summary. Sometimes this is further broken down into external and differential PIOs. The Number of external PIOs is based on the number of PIO pads programmed with a single-ended signal standard out of the total number of bonded pads on the chip die. The Number of differential PIOs counts those PIOs programmed with a differential signal standard.

- Design Errors/Warnings – Shows any warnings or errors encountered by the mapping process. For example, the section will report a pad that is not connected to any logic, or a bidirectional pad that has signals passing only in one direction.

  This section will also report excluded or disabled preferences. These are constraints from the logical preference file (.lpf) that were not implemented for some reason. The name of the .lpf file is given and the applicable components for the excluded preferences.

- IO (PIO) Attributes – Provides a table by input/output port of programmed direction, I/O type, and whether the PIO is registered. Shows any attributes (properties) specified on PIO logic elements.

- Removed Logic – Describes any logic that was removed from the input logical Native Generic Database (.ngd) file when the design was mapped. Logic may be removed for the following reasons:

  - A design uses only part of the logic in a library macro.
  - The design has been mapped even though it is not yet complete.
  - The mapping process has optimized the design logic.
  - Unused logic has been created in error during design entry.

  This section also indicates which nets were merged when a component separating them was removed.

  The Removed Logic section also enumerates how many blocks and signals were removed from the design, including the following kinds of removed logic:

  - Blocks clipped – A "clipped" block is removed because it is along a path that has no driver or no load. Clipping is recursive; that is, if Block A becomes unnecessary because logic to which it is connected has been clipped, then Block A is also clipped.
  - Blocks removed – A removed block is removed because it can be eliminated without changing the operation of the design. Removal is recursive; that is, if a Block A becomes unnecessary because logic to which it is connected has been removed, then Block A is also removed.
  - Blocks optimized – An "optimized" block is removed because its output remains constant regardless of the state of the inputs—for example, an AND gate with one input tied to ground. Logic generating an input to this optimized block (and to no other blocks) is also removed, and appears in this section.

- Signals removed – Signals were removed because they were attached only to removed blocks.
- Signals merged – Signals were combined because a component separating them was removed.

- Memory Usage – Lists all memory instances and all of the device resources that are used; for example, EBRs, RAM SLICEs, Logic SLICEs and PFU registers. Information on data width, type of element used, and attributes is also reported.
- ASIC Components – Lists any ASIC instances in the design by name and specifies the type used for each. These components mostly include processors or memory blocks such as ROM, RAM, and EEPROM.
- Added Logic – Describes any logic that the mapping process added to the design.
- Expanded Logic – Describes the mapping of logic that has been added to the physical design file (.ncd) to resolve blocks.
- Symbol Cross Reference – Shows where symbols in the logical design were mapped in the physical design. By default, Map will not report symbol cross references unless you specify the –xref_sym Map command line option or enable the Report Symbol Cross Reference property for the Map Design process in the Diamond environment.
- Signal Cross Reference – Shows where nets in the logical design were mapped in the physical design. By default, Map will not report symbol cross references unless you specify the -xref_sig Map command line option or enable the Report Signal Cross Reference property for the Map Design process in the Diamond environment.
- DRC Messages – Shows any warnings or errors generated as a result of the design rule tests performed at the beginning of the mapping process. These warnings and errors do not depend on the device to which you are mapping.
- Physical Design Errors and Warnings – Shows errors and warnings associated with problems in assigning components to sites. For example, an attribute on a component specifies that the component must be assigned to site AK, but there is no site AK on the part to which you are mapping. Another example would be a problem in fitting the design to the part where the design maps to 224 logic blocks, but the part only contains 192 logic block sites.
- Schematic Attributes – Shows any attributes (properties) embedded in the input EDIF produced by logic synthesis or a schematic editor. Attributes are translated into preferences in the physical preference file (.prf) that Map produces.

Note

The term “schematic attribute” does not imply that you captured your design in a schematic editor. In the context of mapping or place and route, it refers to those attributes embedded in the input EDIF 2 0 0 netlist as distinct from preferences produced in Diamond’s preference-editing views or the EPIC Device Editor.

- PIO Utilization Report – Provides an account of the exact number and percentage of the total available PIO resources used in the design and breaks down the number of PIOs between differential and non-differential (single ended) I/O component types. It also breaks down the number of DDR cells used, as shown in the following example.

```
Total number of PIOs: 91 out of 222 (41%)
      Number of single ended PIOs: 89 out of 222 (41%)
      Number of differential PIOs: 2 (represented using 1
PIO comps in NCD)
   Number of IDDR/ODDR/TDDR cells used: 26
      Number of IDDR cells:   8
      Number of ODDR cells:  16
      Number of TDDR cells:   2
   Number of PIO using at least one IDDR/ODDR/TDDR: 16 (1
differential)
      Number of PIO using IDDR only:        0 (0
differential)
      Number of PIO using ODDR only:        7 (0
differential)
      Number of PIO using TDDR only:        0 (0
differential)
      Number of PIO using IDDR/ODDR:        7 (0
differential)
      Number of PIO using IDDR/TDDR:        0 (0
differential)
      Number of PIO using ODDR/TDDR:        1 (1
differential)
      Number of PIO using IDDR/ODDR/TDDR:   1 (0
differential)
```

- PLL/DLL Summary – Provides instance name, type, clock frequency, pin/node values, and a listing of all settings for each PLL or DLL in the design.
- OSC Summary – Provides the oscillator's instance name, possible oscillator primitive type, output clock, and nominal frequency. This section is only included in the map report if the OSC primitive has been instantiated in the design. Below is an example of an OSC Summary in a PAR report:

```
OSC Summary
-----------

OSC 1:                                      Pin/Node Value
  OSC Instance Name:                                osc
  OSC Type:                                         OSCF
  OSC Output:                               NODE
reg_clk_2x
  OSC Nominal Frequency:                            130.0
```

- PGROUP Utilization – Reports any PGROUPs that exist in the design and the resources that they occupy in terms of SLICEs within PFUs. Also an account of the size of the bounded area in terms of PFUs are reported and whether the PGROUP is anchored or unbounded (floating).
- Run Time and Memory Usage – Lists total CPU time, real time, and peak memory usage related to the Map run.

PFU Logic Mapping with HGROUP/ UGROUP

When an HGROUP/UGROUP is defined in an EDIF netlist, Map will prevent logical elements of different HGROUPs/UGROUPs to be mapped into the same Programmable Function Unit (PFU). By doing so, a natural partitioning of the mapped design is achieved.

Map then groups all PFUs of the same HGROUP and writes out PGROUP preferences. The HULOC attribute translates to the LOCATE PGROUP preference. The LOC attribute translates to the relative location within a PGROUP preference.

See Also ▶"UGROUPs" on page 393

Packing of Duplicate Registers

Designers often duplicate registers in order to reduce fan-out and improve timing. Duplicated registers are registers with the same data and control paths. The software keeps duplicated registers from being mapped into the same PFU.

If the COMP= attribute is attached to duplicate registers, it will override duplicate register packing.

Map Register Retiming

Register retiming is a technique for improving timing. It involves moving registers across logic to shift path delay from one side of a register to the other side. This retiming technique has the same goal as clock boosting (or cycle stealing), which adjusts the timing by introducing predefined clock delays.

The map register retiming property moves registers across combinational logic to balance timing according to the constraints t_{SU} (INPUT_SETUP), t_{CO} (CLOCK_TO_OUT) and f_{MAX} (FREQUENCY). This is a typical logic optimization technique to balance combinational logic across register-pairs to allow you to maximize clock frequency. However, there is no guarantee that a better f_{MAX} can be achieved. The f_{MAX} constraint activates retiming around all registers. The t_{SU} and t_{CO} constraints may de-activate retiming on I/O registers depending on the balancing of t_{SU} versus f_{MAX} and t_{CO} versus f_{MAX}.

The retiming process stops when it estimates that the constraints have been met.

To enable register retiming in your design:

- Enable the **Register Retiming** Map option in the Strategies dialog box, as explained in “Setting Map Design Options” on page 591. This Map option will be available after an initial design run, not before. Values are True and False (default).

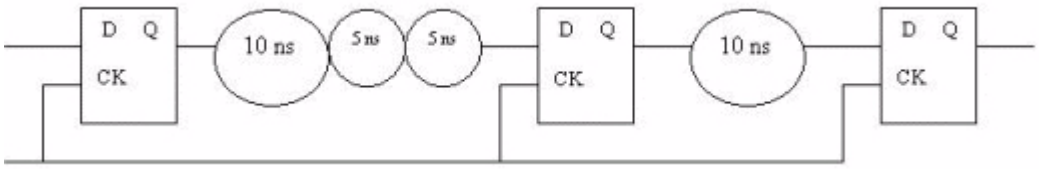
  See “Register Retiming” on page 1142 for details on option settings and default values. This feature can also be prevented for specific design elements with the NORETIME attribute.

- For command line usage, review the documentation on the -retime parameter in the “Running MAP from the Command Line” on page 2495 topic.

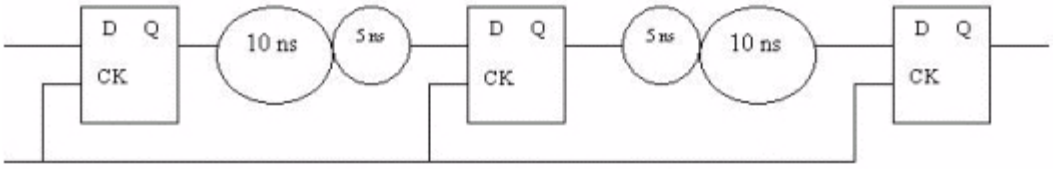
Register retiming can be either forward or backward. Forward retiming will move a set of registers that are the inputs of logic to a single register at its output. Backward retiming will move a register that is at the output of a logic to a set of registers at its input. Retiming works on data path, and has variable delay shift and variable area cost from design to design.

One big positive factor about register retiming is that it will allow for more delay shift. On the negative side, it changes your netlist, making debugging more difficult. It also has a minimum delay shift of one logic level: for example, a LUT.

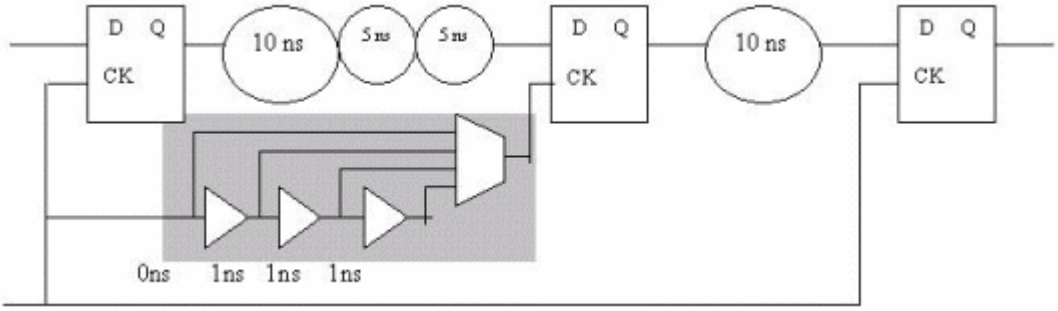
Example: the original network



retiming approach



Clock Boosting approach



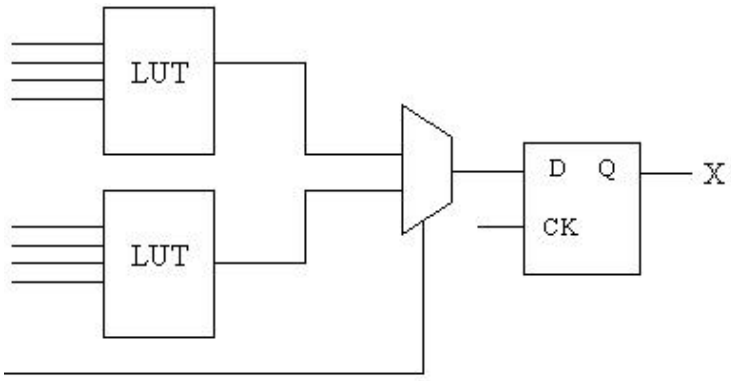
For comparison, clock boosting (only applicable to LatticeSC) works on clock paths and has fixed delay (e.g., 0 ns, 1 ns, 2 ns, or 3 ns) and a fixed area cost (on silicon). The delay shift is accurate after Place & Route (as fine as <= 1ns). Unfortunately, clock boosting requires the use of extra silicon area even if it is not used, and delay shift is limited to few choices up to about 3ns or more. So, please take these considerations into account when you are trying to optimize timing in your design using either of these features.

Note

Like clock boosting, register retiming, in most cases will improve your design's maximum frequency; however, improved timing is design-dependent and is not guaranteed in all cases. Circuits that have considerably unbalanced delay on two sides of a register will benefit the most from register retiming. Circuits that are already register-balanced may receive no benefit at all.

What Should You Retime?

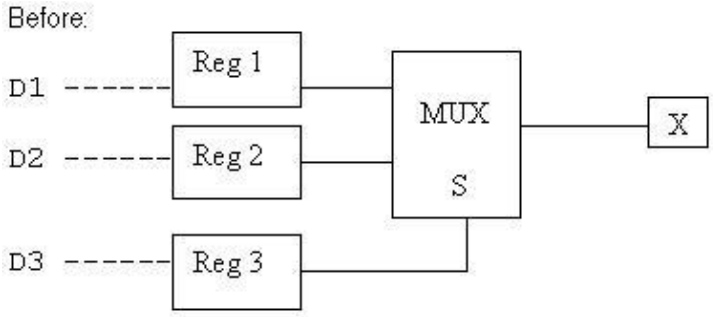
- In most cases, a register can be moved across a combinational logic component that has a known functionality, such as AND/OR gates, LUTs, etc.
- In most cases, registers are moved across one logic component at a time. Special architecture considerations will move registers across multiple levels of combinational logic components such as MUX5 (2 LUTs + 1 MUX2), carry chains, etc.



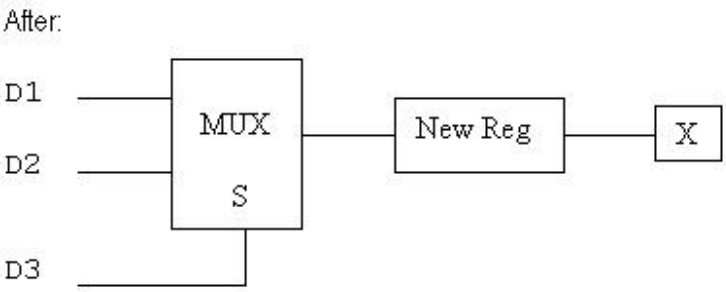
When Should You Retime? This depends upon timing constraints (i.e., f_{MAX}, t_{SU}, and t_{CO}). First compare the slack time on input (D) and output (Q) sides of the register and determine which direction to move.

When to move registers forward: when the output port of a register (Q) is much more critical than its input port (D) and the projected worst slack is better. Forward retiming generally saves area; that is, it reduces register counts.

Legal condition: all input registers must be “compatible”.



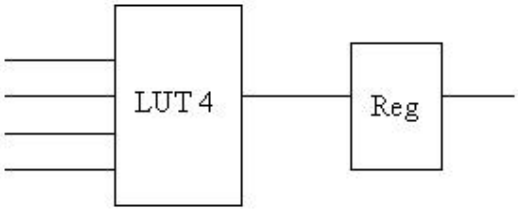
Reg1, 2, 3 must be compatible (i.e., have the same control signals).



When to move registers backward: when the input port of a register (D) is much more critical than its output port (Q) and the projected worst slack is better. The register needs to be replicated to all inputs of the combinational components it moves across.

What are the Costs? Area cost – The primary area cost is due to register replication in backward retiming.

Example: Four registers are replicated



Although forward retiming reduces the number of registers, more than 70 percent of retiming happens in backward retiming. So overall, it results in an area cost. Is this an issue? Statistically, most designs have more than double the amount of LUTs than FFs, because many FFs in used PFUs are left unused. They can be used for retiming purposes. In implementation, we have soft-constrained a certain percentage of device resources to allow for FF replication.

Run time cost – The main computation for retiming occurs in the timing engine. This will contribute to a longer run time in the flow.

NORETIME Attribute You can prevent retiming on an instance and its hierarchy with the NORETIME attribute. The NORETIME IO attribute will globally disallow retiming on I/O registers at the top level.

Preference Support Retiming supports six timing preferences. They are INPUT_SETUP, CLOCK_TO_OUT, FREQUENCY, MULTICYCLE, MAXDELAY, and PERIOD. Other preferences have no effect on retiming.

See Also ▶"Setting Map Design Options" on page 591

▶ "NORETIME" on page 1317 (HDL Attribute)

Place and Route

After a design has undergone the necessary translation to bring it into the physical design (.ncd) format during mapping, it is ready for placement and routing. Placement is the process of assigning the device-specific components produced by the mapping process to specific locations on the device floorplan. After placement is complete, the route phase establishes physical connections to join components in an electrical network. The place and route process takes a mapped physical design (.ncd) file and an optional preference (.prf) file, and places and routes the design. Placement and routing of a design can be cost-based or timing driven.

In most cases, your design will require timing-driven placement and routing, where the timing criteria you specify influences the implementation of the design. Static timing analysis results will show how constrained nets meet or do not meet your timing preferences.

In the Diamond Process view under Place & Route Design, a Place & Route Trace process is available that runs static timing analysis on the place and routed .ncd file. This process reports any timing errors associated with preferences and generates a report. The Place & Route Trace report can be accessed from the Analysis Reports folder of the Reports window.

The I/O Timing Analysis and I/O SSO Analysis processes also generate reports. View the I/O Timing Report to view data such as input and setup times on I/O in worst case scenarios and clock-to-output delay. Use the SSO analysis report to examine simultaneous switching outputs. These reports help ensure that your I/O plan meets the I/O standards and power integrity requirements of the PCB design.

Thermal Analysis Process (MachXO3L/LF 9400C/E devices only). This process must be run for MachXO3L/LF 9400C/E devices before running Bitstream File and JEDEC File processes. The Thermal Analysis process calculates the maximum safe ambient temperature for the user design implemented in MachXO3-9400C and mounted on three standard boards: JEDEC, Small, and Medium boards. This is to assist designers to determine if their design, implemented in MachXO3L/LF 9400C/E device, could safely be used in their application environment.

The thermal analysis software uses the clock specified in the TWR file. If the clock frequency is not set in the TWR file, the thermal analysis process errors out. The designer can either set the clock frequency in the strategy section or in the timing constraint file which will be reported in the TWR file after running Timing Analysis.

The Thermal Analysis report will include Device Operation Conditions, Operating Clock frequency, Percentage of LUT utilization; Percentage of I/O Utilization, and Activity Factor information.

The report contains Thermal Impedance and Maximum Ambient temperature based on three different Board Mounts, No Heat Sink, and No Air Flow.

By default, the Thermal Analysis process is checked. Uncheck this box if you do not wish to run this process.

See Also ▶"Setting Place & Route Design Options" on page 603

- "Options and Processes to Improve PAR Results" on page 604
- "Place & Route Input Files" on page 611
- "Place & Route Output Files" on page 611
- "Running Place & Route Design" on page 612
- "PAR Report File" on page 618

Setting Place & Route Design Options

Placement and routing options can influence the performance and utilization of the design implementation and ease incremental design changes. Some options, such as "Placement Sort Best Run," affect the way the results are reported. You can set Place & Route Design options in the Strategies dialog box in Diamond or use the par command-line program.

To set Place & Route Design options in the Diamond environment:

1. Choose **Project > Active Strategy > Place and Route Design Settings** to open the Strategies dialog.
2. For each option that you want to change, double-click a cell in the Value column.
3. Select the desired value from the drop-down menu, enter text, or click the browse button, as appropriate.

4. Click **OK**.

See Also ▶"Place & Route Design Options" on page 1144

- "Hold-Time Error Correction" on page 605
- "Running PAR from the Command Line" on page 2505
- "PAR Report File" on page 618

Options and Processes to Improve PAR Results

Experimenting with place and route settings in the Strategies dialog box can help improve your placement and routing results. Try changing certain strategy options to help meet your objectives, as in the following scenarios:

- Some number of connections are left unrouted.

  Increase the number of routing passes to help produce a fully routed design.

- Timing period/frequency objectives are not met.

  Try one or more of the following techniques:

  Increase the number of routing passes.

  Increase the effort level of the placer.

  Increase the number of times the placer runs.

- Efforts to meet setup and hold time requirements are increasing run time.

  Make sure that Auto Hold-Time Correction is set to **On**, which is the default setting. This will cause hold time correction to run automatically without increasing run time.

- A small amount of your logic design has changed and a rapid update is desired.

  Using the NCD Guide File option, select a previous version of your physical design (.ncd) file to guide the place and route process.

Other advanced options, such as clock boosting and re-entrant routing, are available from the command line. Clock boosting, (parcb), forces the router to insert extra wires to a Primary or Secondary route clock path to help compensate for hold time violations. Re-entrant routing, which uses the -kp option, instructs PAR to skip placement and leave the existing routing in place.

See Also ▶"Running Place & Route Design" on page 612

- "Hold-Time Error Correction" on page 605
- "Run Time Reduction" on page 609
- "Running PAR from the Command Line" on page 2505
- "Place & Route Design Options" on page 1144

- “Cost-Based Place & Route” on page 630
- “Place & Route Input Files” on page 611
- “Place & Route Output Files” on page 611

Hold-Time Error Correction

Hold-time error correction is a subtask in routing. In previous versions of Lattice Diamond (v3.1 and earlier), a post-step called “auto hold-time correction” (AHC) was carried out after a design was fully routed if there were any hold timing errors. This approach has the following issues:

1. A design could become unrouted after turning feature “AHC” on, especially for congested, hard-to-route designs.
2. Sometimes the performance on the setup could be decreased during the correction of hold timing errors.

A feature called “simultaneous setup/hold timing optimization” (SSH) was introduced in Lattice Diamond v3.2 onward. This feature considers both setup and hold timing at the same time in order to achieve better timing result and address routability. However, the feature “SSH” isn’t available for the routing method “CDR”, and it’s not supported for stand-alone hold timing correction. The table below shows the usage of SSH and AHC features at different settings.

Table 48: Usage of SSH and AHC Features at Different Settings

Command Line Option	Description	Routing Method			
		NBR (default)		CDR	
		SSH	AHC	SSH	AHC
Default	Turn on hold timing error correction only if there are no setup timing errors.	Auto	Off	Off	Auto
-exp parHold=0	Turn off hold timing error correction.	Off	Off	Off	Off
-exp parHold=1	Turn on hold timing error correction with default effort level.	On	Off	Off	On
-exp parHold=2	Turn on hold timing error correction with high effort level.	On	Off	Off	On
-exp parHoldOnly=1	Do hold timing error correction only.	Off	On	Off	On
-exp parHoldAgain=1	Do hold timing error correction again.	Off	On	Off	On

Simultaneous Setup/Hold Timing Optimization Simultaneous Setup/Hold Timing Optimization (SSH) was introduced in Lattice Diamond v3.2. This feature considers both setup/hold timing optimization during the entire routing process. Using the same routing engine and API functions, all the signals are routed under both setup/hold constraints.

For each signal connection, the timing slacks are calculated for both setup and hold constraints. By using slack allocation technology, a delay window is

created for each connection. If each connection can be routed within such delay window, the design will meet both setup/hold timing constraints. A patented routing technique is then called to do routing for the given specific delay window.

Compared to traditional “AHC” feature, the feature “SSH” can help the routing tool to achieve better results regarding hold timing error correction and routability improvement.

When the feature “SSH” is turned on, you should see the following message in par log file:

```
Start NBR section for setup/hold timing optimization with
effort level 3 at <time & date>
```

At the end of par log file, you should see the final timing info similar as following:

```
PAR_SUMMARY::Worst  slack<setup/<ns>> = 0.996
PAR_SUMMARY::Timing score<setup/<ns>> = 0.000
PAR_SUMMARY::Worst  slack<hold /<ns>> = 0.006
PAR_SUMMARY::Timing score<hold /<ns>> = 0.000
```

However, the timing models used in PAR runs may be different than the ones used in TRCE runs, so always use TRCE to verify the timing of your designs.

The feature “SSH” can be disabled by setting “-exp parUseSSH=0” at “Command Line Options” in the Place & Route Strategy window. Once the feature is disabled, the feature “AHC” will be used accordingly.

Auto Hold-Time Correction The Auto Hold-Time Correction (AHC) option causes Place & Route to process setup and hold time requirements in a single flow. The auto hold time correction is performed as soon as the setup timing score of 0 has been achieved. This option improves timing closure without increasing run time. It is supported for all FPGA device families and is set to **On** default.

- What Does Auto Hold-Time Correction Do?

  Auto Hold-Time Correction forces the router to automatically insert extra wires to compensate for hold time violations. The PAR process runs with the setup time target, and AHC begins as soon as a setup timing score of 0 is reached. If a hold time error is found, the hold time correction will kick off automatically. The router will try to resolve any hold time violations on registers that have preferences placed on them, including input I/O to register, register to register, and register to output I/O.

- What if a Setup Timing Score of 0 is not Achieved?

  If a setup timing score of 0 is not achieved, Auto Hold-Time Correction will be skipped during PAR unless AHC has been turned on through the command line.

  Failure to achieve a setup timing score of 0 can happen often, especially in the early design stage. When this happens, you will see a warning message.

```
Setup time error exists in the design, AHC (Auto Hold
Correction) optimization is skipped.
```

You can use the command line to force Auto Hold-Time Correction when setup timing is not achieved.

- From the command line, rerun Place & Route, using the following par option.

  ```
  -exp parHold=1
  ```

- Two Methods for Implementing Auto-Hold Time Correction
  - In the Diamond environment, this feature is implemented as a PAR switch setting for all implementation runs. Auto Hold-Time Correction runs automatically within the normal PAR flow. By default, the Auto Hold-Time Correction option is turned on in the active strategy.
  - You can also run through the flow with this setting turned off and then use the Standalone Auto Hold-Time Correction, which is available from the command line. The standalone program checks the input NCD for hold time violations and corrects any violations it finds.

To set Auto Hold-Time Correction in the Diamond environment:

1. Click **Project > Active Strategy > Place and Route Settings**.
2. In the Strategies dialog box, select **Auto Hold-Time Correction**.
3. Double-click the Value cell and choose **On** or **Off** from the drop-down menu.

To run Auto Hold-Time Correction from the command line:

- Specify the following command:

```
par mapped.ncd output.ncd design.prf
```

To force Auto Hold-Time Correction:

- Specify the following command:

```
par -exp parHold=1 mapped.ncd output.ncd design.prf
```

- To force AHC with extra effort to correct hold time errors, specify the following command:

```
par -exp parHold=2 mapped.ncd output.ncd design.prf
```

To run Stand-alone Hold-Time Correction:

- Specify one of the following example commands:

```
par -kp -w placed.ncd output.ncd design.prf -exp
parHold=1:parHoldOnly=1

par -kp -w placed.ncd output.ncd design.prf -exp
parHold=1:parHoldIter=5:parHoldLimit=500:
parHoldSpeedGrade=6:parHoldOnly=1
```

After the Auto-Hold Time Correction has been set from the command line, it will override the setting in the active strategy. It will force AHC to run if the setup timing score of 0 is not met.

See Also ▶"PAR Explorer Options" on page 2510

- "Running PAR from the Command Line" on page 2505
- PAR Explorer (-exp) Option – parHold
- "Changing Temperature and Voltage for Auto Hold-Time Correction" on page 608
- "Run Time Reduction" on page 609

Changing Temperature and Voltage for Auto Hold-Time Correction

When you enable Auto Hold-Time Correction and run a design flow, all timing is based on the same temperature and voltage settings that are defined in TEMPERATURE and VOLTAGE preferences.

Typically, the software will select the worst case scenarios for both setup and hold time analysis: the lowest voltage and highest temperatures for setup time (slowest time), and the highest voltage and lowest temperatures for hold time (fastest time). However, if you specify your own TEMPERATURE and VOLTAGE preferences, the software will assume that these settings are for setup analysis only.

To change your preferences so that hold time correction only is employed:

1. During your design flow in Diamond, run **Place & Route Design** with Auto Hold-Time Correction turned OFF.

   If you are using the command line, use the option **-exp parHold=0** in your command; for example:

   ```
   par mapped.ncd routed.ncd preference.prf -exp parHold=0
   ```

   This will optimize the maximum delay based on the current temperature/voltage settings defined in preferences, and it will skip hold timing correction.

2. Specify the desired TEMPERATURE and VOLTAGE preference settings in the .prf file.

3. Turn on the Auto Hold-Time Correction, and then rerun **Place & Route Design** to fix the hold timing problem. You define a different temperature/voltage only for hold timing.

   The following example shows how you would use the .ncd file from Step 1 to place and route the design.

```
par -kp routed_input.ncd routed_output.ncd preference.prf -
exp parHold=1:parHoldOnly=1
```

See Also ▶"Running PAR from the Command Line" on page 2505

- PAR Explorer (-exp) Option – parHold
- "Hold-Time Error Correction" on page 605

Run Time Reduction

Run Time Reduction is a multi-seed PAR option that uses checkpoints for comparing the result of the current placement seed with the results already obtained. A placement seed that is not producing better results will automatically be terminated. Beginning with Diamond 2.1, the Runtime Reduction option is ON by default.

As an example, let's say that you are doing a multi-seed run and you want to run Place & Route with ten seeds and save the NCD files for three of them. The PAR process will do the first three seeds as normal runs. But starting from the fourth seed, PAR will check the timing result in terms of the worst slack at different checkpoints and will compare it to the results already obtained. If the worst slack is worse, the remainder of the routing tasks for the fourth placement seed will be skipped, and PAR will continue to the next seed. Clearly, if the number of seeds to be saved is equal to the number of runs, there will be no run time reduction; and saving just one seed will produce the biggest run time reduction.

When a placement seed has been terminated, the following type of message will appear in the .par file for the 5_x placement seed:

```
************************************************************************
WARNING: The worst setup slack (-7.302ns) after initial routing exceeds the specified
threshold -5.164 ns. This par run will be terminated.
************************************************************************


!!!WARNING!!! Catch an exception in router NBR.
Reason : BASRT::Exceed the initial slack threshold -5.164ns.
Func   : RT_PFSTAR::routeUnifyPrepCongestion

!!!EXCEPTION!!! - BASRT::Exceed the initial slack threshold -5.164ns.
```

The PAR report file for the project, which you can access from the Reports tab in Diamond, will include a Worst Slack column in the Cost Table Summary, as shown in the following example:

```
Cost Table Summary
Level/       Number       Worst        Timing       Run          NCD
Cost [ncd]   Unrouted     Slack        Score        Time         Status
----------   --------     -----        --------     -----        ------
5_1    *     0            -3.038       485572       01:11        Complete
5_5    *     0            -3.238       476719       01:12        Complete
5_4          0            -4.446       564958       58           Complete
5_2          0            -5.164       729873       56           Complete
5_3          -            -            -            57           Skip

* : Design saved.
```

In the Cost Table Summary, the third column shows the worst slack (setup) after routing for the different seeds. Notice that seed 3 is skipped; therefore, there is no saved .ncd file or related timing information for seed 3.

By default, worst slack is used to compare the current seed to the best result obtained so far. However, you can change this so that the cost table is sorted based on timing score. In Diamond, choose **Project > Active Strategy > Place & Route Design Settings** and change the value for Placement Sort Best Run.

You can turn the Run Time Reduction feature off from the command line, with the following command:

```
-exp parASE=0
```

You can also turn this feature off in the Active Strategy user interface. In the Place & Route Design section, select the Command Line Options row and type **`-exp parASE=0`** in the Value cell.

See Also ▶"Multiple PAR Iterations" on page 612

- "Hold-Time Error Correction" on page 605
- "PAR Report File" on page 618

Clock Skew Minimization

Clock Skew Minimization minimizes clock skew on selected non-global clock nets that warrant optimization. It provides pre-computed delay that is lower bound, which provides balanced routing.

This Clock Skew Minimization feature uses a program called CSM that will attempt to optimize the timing on nets you specify in the preference file before rerunning PAR. Details for using this feature are provided in application notes.

To turn on clock skew minimization in the Diamond environment:

1. Choose **Project > Active Strategy > Place & Route Design Settings**.
2. In the Strategies dialog box, select **Clock Skew Minimization**.
3. Double-click the Value cell and choose **1** or **2** from the drop-down menu.

   The “1” setting routes from driver pins to each clock load.

   The “2” setting routes a clock trunk first and then routes from the trunk to each clock load.
4. Click **OK**.

See the PAR -exp clockSkewMin Option for command-line instructions.

See Also ▶“Running PAR from the Command Line” on page 2505

Place & Route Input Files

PAR uses the following input files:

- Native Circuit Description File (.ncd) – This file is a physical description of the design in terms of the components in the target device.
- Physical Preference File (.prf) – This ASCII text file is an output of the Map Design process and includes preferences that were specified during design entry. It typically contains timing constraints that were specified in the logical preference file (.lpf). Preferences can indicate such things as clock speed for input signals, the external timing relationship between two or more signals, absolute maximum delay on a design path, or a general timing requirement for a class of pins.

  Editing the .prf file is not recommended. You should only edit constraints in the logical preference (.lpf) file or the HDL source to ensure the repeatability of the design flow through logical and physical domains. You can edit logical preferences using a text editor or Diamond’s preference-editing views.

See Also ▶“Place and Route” on page 602

- “Place & Route Output Files” on page 611
- “Setting Preferences” on page 459

Place & Route Output Files

The following files are the output files generated by the Place & Route Design process:

- Native Circuit Description File (.ncd) file — a placed and routed physical design file, which might contain placement and routing information in varying degrees of completion. One or more .ncd files will be produced depending on the options you choose. See “Multiple PAR Iterations” on page 612.

- PAR Report File (.par) — a PAR report including summary information of all placement and routing iterations.
- PAD Specification File (.pad) — a report file containing I/O pin assignments.
- Delay Report File (.dly) — an optional ASCII text file that includes a delay analysis of the 20 nets with the longest delay. PAR generates this file when the “Create Delay Statistic File” has been set to True in the Strategies dialog box or when the -y option is used in the command line.

  The option also appends a Delay Summary Report to the end of the Place & Route report. The first column of this summary gives the actual averages for the design. The figures in the second column, which are enclosed by parentheses, indicate a “worst case” scenario for the delay.

See Also “Place and Route” on page 602

- “Place & Route Input Files” on page 611

Running Place & Route Design

In the Diamond environment, the Place & Route Design process automatically assigns device-specific components to locations and connects them.

To run Place & Route Design in the Diamond environment:

- In the Process view, double-click **Place & Route Design**.
- Alternatively, right-click Place & Route Design and choose **Run** from the pop-up menu.

See Also “Setting Place & Route Design Options” on page 603

- “Running PAR from the Command Line” on page 2505
- “Running Processes” on page 70

Multiple PAR Iterations

The “Place Iterations” option for Place & Route Design sets the maximum number of placement/routing passes.

If you specify options that produce a single output design file, your output consists of a single .ncd file, .par file, and .pad file. The .par and .pad files all have the same root project name as the .ncd file. Optionally, you can output a Delay Statistics (.dly) file that will also, by default, assume the project name. See “Delay Statistics Report File” on page 630 for details.

If you run multiple placement and routing iterations, you produce a .ncd, .par, .dly, and .pad file for each iteration. After each iteration, the program saves the physical design (.ncd) file for every instance where the timing core has improved. This means that during long PAR runs, the current .ncd file with a minimal timing score is always available. This behavior prevents a few rare

cases where timing scores may have gone up rather than down when performing multiple PAR iterations. The timing score is determined as a weighted sum of several parameters including the number of unrouted nets, number of timing constraints not met, amount by which the timing constraints were not met, maximum delay on nets, and maximum delay on the ten highest nets.

As the par command is performed, PAR records one .par file (a summary of all placement and routing iterations) at the same level as the directory you specified. It also places within the directory an .ncd, .par, .dly and .pad file for each individual iteration.

The file names for the output files use the naming convention <*effort-level*>_<*cost-table-entry*>; for example, 1_1.ncd, 1_1.par, 1_1.dly, and 1_1.pad. In this example, the effort level and cost table entries start at 1 (the default effort level is 5).

Note

If you wish to further optimize your place-and-route results beyond multiple PAR iterations, a process which uses timing as the sole criteria for optimization, see "Using MPARTRCE to Optimize PAR Iterations" on page 613. In addition to timing score, the MPARTRCE tool's methodology uses a performance score when it determines the best design run for your purposes.

See Also

- "Place and Route" on page 602
- "Using MPARTRCE to Optimize PAR Iterations" on page 613
- "Run Time Reduction" on page 609

Using MPARTRCE to Optimize PAR Iterations

MPARTRCE is a multi-PAR program that uses a performance score for selecting the best iteration. It is available from the command line.

Multi-PAR When PAR processes design iterations using different placement seeds to produce the best possible placed-and-routed design, it does so by performing and evaluating each design on a timing score basis. See "Multiple PAR Iterations" on page 612.

The timing score is a useful measure for algorithms to perform optimal place-and-route on the design. However, timing score is not necessarily the best criteria to use to select the best iteration with respect to the user's preference. Furthermore, the timing score becomes zero if all the constraints are met, making the choice of the best implementation random because multiple implementations meet the timing constraints.

In multiple PAR scenarios, you are interested in the implementation that results in the best timing performance. For example, in a single clock design, where the setup and clock-to-out times are not critical, the best iteration is the one that results in the highest f_{MAX} in static timing analysis. When there are more timing preferences, the best result would be the one that scores best for each of the timing preferences.

MPARTRACE Multi-PAR The multiple PAR of MPARTRCE continues to use the timing score for PAR algorithms in multiple PAR iterations, but it also employs a measure called performance score for selecting the best iteration.

MPARTRCE attempts to do the following:

- Use the existing method for running multiple PAR iterations, each using a different placement seed and using “timing score” criteria for optimizing place and route processes.
- Automatically run static timing analysis (TRCE) on the implementation files (.ncd) after each PAR iteration.
- Establish a new selection criterion to pick the iteration that best satisfies user preferences and requirements.
- Provide options to specify special requirements, preferences, and criteria for selecting the best result.

Performance Score The MPARTRCE method for selection of the best design uses a performance score in its determination. The Performance Score (PS) is defined as the following:

PS = FS(i)

where "i" is the number of preferences used for PS computation and FSi is the Frequency Score for the ith preference. The Frequency Score (FS) is defined below:

If, (Achieved_time)i < (Required_time)i ,

FSi = (Required time - Achieved_time) * wi

else,

FSi = (Required time- Achieved_time) * pi

where wi is the weight factor for the ith preference and pi is the penalty factor for the ith preference. The weight factor is associated with positive slack (timing preference met) and penalty with negative slack (timing preference violated).

The performance score is calculated for each iteration’s implementation result (.ncd file), and the one with the highest (algebraic maximum) performance score is selected as the best result.

Preferences and achieved results that are in frequency domain will be converted to the corresponding time domain values and used in performance score equations.

MPARTRCE Process MPARTRCE executes the following operations:

1. Reads in the constraints from the physical preference file (.prf) and determines the weights and penalties based upon a specific .ini file and user overrides.

2. Runs PAR, TRCE and result selection algorithms multiple times. During result selection, it parses the TRACE report for the values of interest, determines the frequency and performance scores, and creates a report in .csv format.
3. After all the iterations, copies the implementation and report files corresponding to the best result from the results directory into the project directory and creates the log file.

If you break the process after any iteration N, the results and reports correspond to the best implementation to that point. The outputs would be equivalent to what you would have obtained if you had run multiple PAR to N iterations.

See Also ▶"Specifying Preferences for MPARTRCE" on page 615

- "Running MPARTRCE from the Command Line" on page 2522
- "Multiple PAR Iterations" on page 612

Specifying Preferences for MPARTRCE

To direct MPARTRCE to perform its intended function, you must accurately guide the tool by setting your timing preferences properly in the PAR physical preference (.prf) file. You can edit preferences in your text editor of choice.

If some of the preferences in the .prf file are not required for result selection or if you must change the values in the file for result selection, a different preference file can be specified for TRACE (trce) and result selection purposes.

In the MPARTRCE command line, this mpartrce.prf file can be named anything and supplied using the **-**tpr option. The mpartrce.prf file can contain any subset of the PAR preference file. If mpartrce.prf exists and is specified in the command line, then that file is used for trce and result selection purposes. Only the timing preferences and their values in the mpartrce.prf file will be used for performance score computation and result selection. In GUI mode, the preference file will have to be named "mpartrce.prf" and must reside in the project directory. The GUI option is not currently supported.

Overriding weights and penalties Performance score computation uses different penalty and weight factors for each timing preference. Default weights and penalties are defined for each preference group (f_{MAX}, t_{PD}, t_{SU} and t_{CO}) in the mpartrce program. It is possible for the advanced users to override these defaults by individually specifying relative weights and penalties for each timing preference line.

The weight and penalty values are specified as comments at the end of each preference line. A comment symbol (#) is placed at the end of the preference line and is followed by the lowercase letter w for weight and the value for weight. Similarly the penalty factor is entered by the lowercase letter p followed by the actual penalty factor value. The preferences that do not have overriding w or p value will use the default values.

The following sample preference lines and the corresponding weight/penalty override values illustrate the override mechanism.

```
FREQUENCY NET "clk_tx" 200.000000 MHz ; # w 20 p 400
  New weight for clk_tx = (default weight) * 20
  New penalty for clk_tx = (default penalty) * 400

FREQUENCY PORT "clk_rx" 150.000000 MHz ; # p 100
  New weight for clk_rx = (default weight)
  New penalty for clk_rx = (default penalty) * 100

INPUT_SETUP ALLPORTS 3.000000 ns CLKPORT "clk" ; # w 0.5 p 2
  New weight for Tsu for clk = (default weight) * 0.5
  New penalty for Tsu for clk = (default penalty) * 2

CLOCK_TO_OUT ALLPORTS 7.000000 ns CLKPORT "clk" ; # w 2
  New weight for clk_tx = (default weight) * 2
  New penalty for clk_tx = (default penalty)
```

The w/p overrides can be specified either in the PAR preference (.prf) file or the timing preference (.tpf) file. The additional preference values specified do not affect the PAR results as they are used only by trce and in the result selection processes. The weights and preference overrides in either preference files do not affect PAR or TRACE programs as they appear as comments for those programs.

Note

Any manual change to the PAR preference file will be lost when a MAP is re-run on the design.

Preference file and override options The following are the different options that can be used:

- Use only the PAR preference file (<proj_name>.prf) and specify no overrides. Here are some things to consider when using this option:
  - You do not need to create additional files or edit the .prf file.
  - You have to rely on over-constrained preference values.
  - You cannot remove an unneeded preference line.
  - You cannot override the default priorities for timing preferences.
- Use only the PAR preference file (<proj_name>.prf), but manually add w/p overrides. Here are some things to consider when using this option:
  - You do not need to create additional files.
  - You have to rely on over-constrained preference values.
  - You cannot remove an unnecessary preference line.
  - You can override the default priorities for timing preferences by specifying w and p. Indirectly, this can eliminate the effect of retaining unnecessary preferences on the performance score.

- The PAR preference file will get overwritten if you re-run map, and all the w/p overrides appearing as comments will be removed.

- Manually create a Timing preference (.tpf) file and specify no overrides. Here are some things to consider when using this option:
  - The Timing preference (.tpf) file can be easily created by copying the PAR preference file. Map can be re-run without affecting this preference file.
  - Over-constrained preference values can be replaced with reasonable values required for result selection.
  - Unwanted preferences can be removed.
  - You have to be content with default weight and penalty factors.
- Manually create a Timing preference (.tpf) file and add w/p overrides to the preferences in it. Here are some things to consider when using this option:
  - The Timing preference (.tpf) file can be easily created by copying the PAR preference file. MAP can be re-run without affecting this preference file or the w/p overrides.
  - Over-constrained preference values can be replaced with reasonable values required for result selection.
  - Unwanted preferences can be removed.
  - Weight and penalty values can be overridden as required to have complete control on the result selection method

See Also ▶"Running MPARTRCE from the Command Line" on page 2522

- "Using MPARTRCE to Optimize PAR Iterations" on page 613
- "Multiple PAR Iterations" on page 612

Place & Route Report Files

The Reports view in Diamond provides immediate access to reports produced by the Place & Route Design process, including the Place & Route report (.par) and the Pad Specification file (.pad). These reports open in the Report view in HTML format.

If you have enabled the Create Delay Statistics File option for the active strategy, a Delay Statistics Report File (.dly) will be included in your project directory. When you run multiple iterations of Place & Route, a .par file, .pad file, and .dly file will be included in your project directory for each run. You can open these files with a text editor.

See Also ▶"Multiple PAR Iterations" on page 612

- "Run Time Reduction" on page 609
- "PAR Report File" on page 618

PAR Report File

The PAR Report (.par) file contains execution information and shows the steps taken as the process converges on a placement and routing solution. To access this file in the Diamond environment, use the Report window.

To view the PAR Report file in Diamond:

- In the Design Summary pane of the Reports window, select **Place & Route**. Your report will appear in the pane to the right.
- Alternatively, use a text editor to open the .par ASCII file from your project directory.

Some of the sections included in a given .par file are described below, with links to a sample .par file. These sections will vary, depending on the FPGA architecture.

Cost Table Summary This section summaries the results of each placement seed and shows the worst slack and the timing score for both setup and hold. For a typical design, using five placement seeds with the best two runs saved, the cost table will resemble the following example:

```
Level/      Number    Worst    Timing   Worst       Timing      Run    NCD
Cost [ncd]  Unrouted  Slack    core     Slack(hold) Score(hold) Time   Status
----------  --------  -----    -----    ----------- ----------- -----  ------
5_1   *     0         0.531    0         0.235       0           26    Complete
5_3   *     0         0.398    0         0.235       0           24    Complete
5_2         0         0.174    0         0.175       0           22    Complete
5_4         -         -        -         -           -           21    Skip
5_5         -         -        -         -           -           18    Skip
```

However, if setup timing is not met, or if the hold time fix is not turned on (see “Auto Hold-Time Correction” on page 1145 and PARHOLD), then the hold time information will not be reported, as shown by the following example:

```
Level/      Number    Worst    Timing   Worst       Timing      Run    NCD
Cost [ncd]  Unrouted  Slack    Score    Slack(hold) Score(hold) Time   Status
----------  --------  -----    ------   ----------- ----------- -----  ------
5_3   *     0         -0.507   6583     -           -           19     Complete
5_2   *     0         -0.700   8195     -           -           20     Complete
5_5         0         -0.822   13747    -           -           20     Complete
5_4         0         -0.869   16325    -           -           20     Complete
5_1         0         -0.914   23897    -           -           24     Complete
```

By default, the Cost Table Summary is sorted by worst slack for multiple placement seeds, as follows:

1. NCD status (completed or not)
2. worst slack (setup)
3. timing score (setup)
4. worst slack (hold)

5. timing score (hold)
6. seed #.

If you would rather that the cost table be sorted by timing score for multiple placement seeds, select the Timing Score option for “Placement Sort Best Run” in the Place & Route Design section of the active strategy. The table will then be sorted as follows:

1. NCD status (completed or not)
2. timing score (setup)
3. worst slack (setup)
4. timing score (hold)
5. worst slack (hold)
6. seed #

Device Utilization Summary This section summarizes the number and percentage utilization of device resources, including I/O, logic, and global signals. Except for PIOs, a single device SITE resource is used for each design COMP of the Native Circuit Description (.ncd) database. For PIOs, twice the number of PIO SITE resources are required for differential I/Os. A single differential PIO COMP uses two PIO SITE resources, whereas non-differential PIO COMPs use one PIO SITE resource.

The PIO report summary counts the number of bonded and unbonded PIOs required to implement single-ended and differential signal standards. PIO utilization reflects the preliminary results reported by the design mapper (map). See the I/O Usage Summary section for details on final results. Utilization reported by Map might be lower, because some SITE resources, such as VREF assignments, are made automatically by PAR.

Report Syntax:

SITE TYPE <# *sites used*> / <# sites of type available on device> <x>% used

Consider the example below. The first column identifies the device type. The second column provides the actual number of available sites versus those used for that device type, and the third column provides the percentage of

available sites used. For PIOs, you will see a breakdown of available versus used bonded pads and a percentage:

```
Device utilization summary:

   APIO                24/36           66% used
   GSR                  1/1           100% used
   IOLOGIC            110/506          21% used
   PIO (prelim)       194/504          38% used
                      194/372          52% bonded
   DQSBUF               8/16           50% used
   EBR                135/225          60% used
   DQSDLL               1/2            50% used
   MULT18               1/88            1% used
   PCS                  2/2           100% used
   SLICE          17827/23832          74% used

   PLL                  3/8            37% used
```

For total PIO resources, a preliminary (prelim) report shows that out of a possible 504 total PIO resources on the device, the design uses 194 or about 38 percent. The following line shows that 194 bonded out pads are being used out of a possible 372 bonded I/Os, which is about 52 percent.

The term “used” means that these I/O are in the design and will be programmed on the device. This report is considered preliminary (prelim) because VREF assignments have not yet been placed. The difference in resource utilization is accounted for later in the PAR report in the I/O Usage Summary (final).

An I/O is considered “bonded” when the packaging of the chip connects the bond pad to a pin on the package.

Note

Bonded pad availability is based on packaging, which varies within a device family from part to part. See the Package Diagrams documentation in addition to the Product Selector Guide on the Lattice web site for details when selecting a device and package for your application.

The rest of the example report shows IOLOGIC, PLL, SLICE and other various resource type usage. As utilization percentage approaches 100 percent on logic or any specific key resource types reported in this section, you might want to consider a larger device that can accommodate better design performance.

Placement This section provides a log of messages produced during the placement phase.

Clock Report This section provides a complete listing of all of the clocks used in the design. Aside from the example, the Clock Summary report would reflect a scenario where only global clocking is used as follows:

```
Quandrants All (TL, TR, BL, BR) - Global Clocks
   PRIMARY  : 1 out of 4 (25%)
     DCS    : 1 out of 2 (50%)
  SECONDARY : 0 out of 4 (0%)
```

I/O Usage Summary (final) This section Summarizes the final number and percentage utilization of device resources, including I/O, logic, and global signals.

Report Syntax:

```
I/O Bank Usage Summary (final):
   <#PIO sites used> out of <#PIO sites of device> (x%) PIO
   sites used.
<#PIO sites used> out of <#PIO sites of device/pkg> (x%) bonded
PIO sites
used.
Number of PIO comps: <#PIO comps which are in the NCD>; <#PIO
comps in NCD which represent differential IO>
Number of Vref pins used: <#VREF pins>
```

Routing This section provides a log of messages produced during the routing phase.

Completion This section shows statistics of the run, including time to route, number and percentage of successfully routed connections, errors and warnings, and final timing score.

Notes regarding the sample place and route report:

- Placer score is a rating of the relative “cost" of a placement. A lower score indicates a better, less costly placement.
- Timing score will always be 0 (zero) if all timing preferences have been met. If not, the figure will be other than 0. This tells you immediately whether your timing preferences have been met.
- Underneath the “Completed router resource pre-assignment” line, warning messages might appear, indicating that a clock signal can suffer excessive delay or skew due to conflicts in clock routing resources.

  For example, the placer will avoid placing two clock signals in the same PIC pair that share the same clock spine. If more than one clock signal are hard-located in the same PIC pair that share a common clock spine, the router will write out a warning message in the PAR Report File (.par) at this point.

- The “Starting iterative routing” section contains figures in parentheses (125818). This represents the timing score for the design (not to be confused with the PAR score) at the end of the particular iteration. When the timing score reaches 0 (as it does in this example after iteration 2), this means that all timing preferences have been met. This timing score (0) also appears at the end of the Delay Summary Report section. Note that the Delay Summary Report is only generated when you use enable the “Create Delay Statistic File” option in the Strategies dialog box or use the -y option in the command line.

  The timing score at the end of the “Starting iterative routing” section might not agree with the timing score at the end of the Delay Summary Report. This can occur if a MAXSKEW preference is scored and not met. The score shown in the Delay Summary Report section will always be the correct one.
- Sometimes the design will be completely routed but the router continues to route in the attempt to adhere to timing preferences.
- There is a section before the “Starting Constructive Placer” message that will list all selected primary clocks.
- When you specify the -y option on the command line, the last section of the .par file summarizes the delay information for the routed design. The first column of this section gives the actual averages for the design. The figures in the second column, which are enclosed by parentheses, indicate a "worst case" scenario for the delay. The PAR run also produces a .dly (delay) file that contains more detailed timing information.

Note

Timing scores can rise when attempting to meet timing constraints, because the router might move a signal to a less favorable path to make a resource available for another signal.

See Also ▶“Place & Route Output Files” on page 611

- “Sample PAR Report File” on page 622

Sample PAR Report File

The following is an example of a typical PAR Report (.par) file. For explanations of each section, see “PAR Report File” on page 618.

```
PAR: Place And Route Diamond Version 3.0.0.41.
Copyright (c) 1991-1994 by NeoCAD Inc. All rights reserved.
Copyright (c) 1995 AT&T Corp.   All rights reserved.
Copyright (c) 1995-2001 Lucent Technologies Inc.  All rights
reserved.
Copyright (c) 2001 Agere Systems   All rights reserved.
Copyright (c) 2002-2013 Lattice Semiconductor Corporation,  All
rights reserved.

Thu Jun 06 12:12:02 2013
```

```
                        C:/lscc/diamond/2.1/ispfpga\bin\nt\par -f
                        attributes_attributes.p2t
                        attributes_attributes_map.ncd attributes_attributes.dir
                        attributes_attributes.prf -gui

                        Preference file: attributes_attributes.prf.


Cost Table Summary
Level/      Number    Worst   Timing  Worst       Timing      Run    NCD
Cost [ncd]  Unrouted  Slack   core    Slack(hold) Score(hold) Time   Status
----------  --------  -----   -----   ----------- ----------- -----  ------
5_1   *     0         0.531   0        0.235       0           26    Complete
5_3   *     0         0.398   0        0.235       0           24    Complete
5_2         0         0.174   0        0.175       0           22    Complete
5_4         -         -       -        -           -           21    Skip
5_5         -         -       -        -           -           18    Skip

* : Design saved.


                        Total (real) run time for 5-seed: 1 mins 42 secs

                        par done!

                        Lattice Place and Route Report for Design
                        "attributes_attributes_map.ncd"
                        Thu Jun 06 12:12:02 2013

                        Best Par Run
                        PAR: Place And Route Diamond Version 3.0.0.41.
                        Command Line: par -w -l 5 -i 6 -t 1 -c 0 -e 0 -gui -exp
                        parUseNBR=1:parCDP=auto:parCDR=1:parPathBased=OFF
                        attributes_attributes_map.ncd attributes_attributes.dir/5_1.ncd
                        attributes_attributes.prf
                        Preference file: attributes_attributes.prf.
                        Placement level-cost: 5-1.
                        Routing Iterations: 6

                        Loading design for application par from file
                        attributes_attributes_map.ncd.
                        Design name: top
                        NCD version: 3.2
                        Vendor:      LATTICE
                        Device:      LFXP2-17E
                        Package:     FTBGA256
                        Performance: 5
                        Loading device for application par from file 'mg5a50x47.nph' in
                        environment: C:/lscc/diamond/3.0/ispfpga.
                        Package Status:                     Final         Version 1.63
                        Performance Hardware Data Status:   Final         Version 10.6
                        License checked out.

                        Ignore Preference Error(s):  True

                        Device utilization summary:
                           GSR                1/1           100% used
                           IOLOGIC           21/364           5% used
                           PIO (prelim)      46/358          12% used
```

```
                     46/201          22% bonded
   SLICE            125/8280          1% used

Number of Signals: 321
Number of Connections: 892
The following 2 signals are selected to use the primary clock
routing resources:
    dec_pll_clk_c (driver: dec_pll/PLLInst_0, clk load #: 53)
    enc_pll_clk_c (driver: enc_pll/PLLInst_0, clk load #: 19)

No signal is selected as DCS clock.

No signal is selected as secondary clock.

Signal rst_n_c is selected as Global Set/Reset.

Starting Placer Phase 0.
.............
Finished Placer Phase 0.  REAL time: 3 secs

Starting Placer Phase 1.
.................
Placer score = 45266.
Finished Placer Phase 1.  REAL time: 14 secs

Starting Placer Phase 2.

Placer score = 44822
Finished Placer Phase 2.  REAL time: 14 secs


Clock Report

Global Clock Resources:
  CLK_PIN    : 0 out of 8 (0%)
  PLL        : 2 out of 4 (50%)
  CLKDIV     : 0 out of 2 (0%)

Quadrants All (TL, TR, BL, BR) - Global Clocks:
  PRIMARY "dec_pll_clk_c" from CLKOP on comp "dec_pll/
PLLInst_0" on PLL site "LLPLL", clk load = 53
  PRIMARY "enc_pll_clk_c" from CLKOP on comp "enc_pll/
PLLInst_0" on PLL site "ULPLL", clk load = 19

  PRIMARY  : 2 out of 8 (25%)
     DCS   : 0 out of 2 (0%)
  SECONDARY: 0 out of 4 (0%)

Edge Clocks:
  No edge clock selected

--------------- End of Clock Report ---------------

I/O Usage Summary (final):
   46 out of 358 (12.8%) PIO sites used.
   46 out of 201 (22.9%) bonded PIO sites used.
   Number of PIO comps: 46; differential: 0
   Number of Vref pins used: 0
```

```
I/O Bank Usage Summary:
+----------+----------------+------------+------------+------------+
| I/O Bank | Usage          | Bank Vccio | Bank Vref1 | Bank Vref2 |
+----------+----------------+------------+------------+------------+
| 0        | 1 / 28 (  3%)  | -          | -          | -          |
| 1        | 0 / 22 (  0%)  | -          | -          | -          |
| 2        | 22 / 26 ( 84%) | 2.5V       | -          | -          |
| 3        | 19 / 24 ( 79%) | 2.5V       | -          | -          |
| 4        | 0 / 26 (  0%)  | -          | -          | -          |
| 5        | 1 / 24 (  4%)  | -          | -          | -          |
| 6        | 2 / 27 (  7%)  | 2.5V       | -          | -          |
| 7        | 1 / 24 (  4%)  | 2.5V       | -          | -          |
+----------+----------------+------------+------------+------------+


                    DSP Utilization Summary:
                    --------------------------------------
                    DSP Block #:              1 2 3 4 5
                    # of MULT36X36B
                    # of MULT18X18B
                    # of MULT18X18MACB
                    # of MULT18X18ADDSUBB
                    # of MULT18X18ADDSUBSUMB
                    # of MULT9X9B
                    # of MULT9X9ADDSUBB
                    # of MULT9X9ADDSUBSUMB

                    Total placer CPU time: 13 secs

                    Dumping design to file attributes_attributes.dir/5_1.ncd.

                    0 connections routed; 892 unrouted.


                    Starting router resource preassignment

                    Completed router resource preassignment. Real time: 19 secs

                    Start NBR router at 12:12:21 06/06/13

                    ************************************************************
                    Info: NBR allows conflicts(one node used by more than one
                    signal)in the earlier iterations. In each iteration, it tries
                    to solve the conflicts while keeping the critical connections
                    routed as short as possible. The routing process is said to
                    be completed when no conflicts exist and all connections are
                    routed.
                    Note: NBR uses a different method to calculate timing slacks.
                    The worst slack and total negative slack may not be the same as
                    that in TRCE report. You should always run TRCE to verify
                    your design. Thanks.
                    ************************************************************

                    Start NBR special constraint process at 12:12:21 06/06/13
```

```
7(0.00%) conflicts; 707(79.26%) untouched conns; 0 (nbr) score;
Start NBR section for initial routing
Estimated worst slack/total negative slack: 0.662ns/0.000ns;
real time: 19 secs
Level 2, iteration 1
71(0.01%) conflicts; 575(64.46%) untouched conns; 0 (nbr)
score;
Estimated worst slack/total negative slack: 0.880ns/0.000ns;
real time: 19 secs
Level 3, iteration 1
95(0.01%) conflicts; 116(13.00%) untouched conns; 0 (nbr)
score;
Estimated worst slack/total negative slack: 0.862ns/0.000ns;
real time: 20 secs
Level 4, iteration 1
47(0.01%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.862ns/0.000ns;
real time: 20 secs

Info: Initial congestion level at 75% usage is 0
Info: Initial congestion area  at 75% usage is 0 (0.00%)

Start NBR section for normal routing
Level 1, iteration 1
41(0.01%) conflicts; 11(1.23%) untouched conns; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.862ns/0.000ns;
real time: 20 secs
Level 1, iteration 2
39(0.01%) conflicts; 16(1.79%) untouched conns; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.862ns/0.000ns;
real time: 20 secs
Level 2, iteration 1
30(0.00%) conflicts; 19(2.13%) untouched conns; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.862ns/0.000ns;
real time: 20 secs
Level 3, iteration 1
24(0.00%) conflicts; 7(0.78%) untouched conns; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.862ns/0.000ns;
real time: 20 secs
Level 4, iteration 1
19(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.757ns/0.000ns;
real time: 21 secs
Level 4, iteration 2
11(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Level 1, iteration 1
Estimated worst slack/total negative slack: 0.482ns/0.000ns;
real time: 21 secs
Level 4, iteration 3
5(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 4
1(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 5
2(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
```

```
Estimated worst slack/total negative slack: 0.482ns/0.000ns;
real time: 21 secs
Level 4, iteration 6
3(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.482ns/0.000ns;
real time: 21 secs
Level 4, iteration 7
3(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 8
1(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 9
1(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.482ns/0.000ns;
real time: 21 secs
Level 4, iteration 10
2(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.482ns/0.000ns;
real time: 21 secs
Level 4, iteration 11
1(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 12
2(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 13
2(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.700ns/0.000ns;
real time: 21 secs
Level 4, iteration 14
1(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.700ns/0.000ns;
real time: 21 secs
Level 4, iteration 15
1(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 16
2(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.569ns/0.000ns;
real time: 21 secs
Level 4, iteration 17
2(0.00%) conflicts; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.570ns/0.000ns;
real time: 21 secs
Level 4, iteration 18
1(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.570ns/0.000ns;
real time: 21 secs
Level 4, iteration 19
0(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.531ns/0.000ns;
real time: 21 secs
```

```
Start NBR section for re-routing
Level 4, iteration 1
0(0.00%) conflict; 0(0.00%) untouched conn; 0 (nbr) score;
Estimated worst slack/total negative slack: 0.531ns/0.000ns;
real time: 21 secs

Start NBR section for post-routing

End NBR router with 0 unrouted connection

NBR Summary
-----------
  Number of unrouted connections : 0 (0.00%)
  Number of connections with timing violations : 0 (0.00%)
  Estimated worst slack : 0.531ns
  Timing score : 0
-----------
Notes: The timing info is calculated for SETUP only and all
PAR_ADJs are ignored.

Hold time optimization iteration 0:
All hold time violations have been successfully corrected in
speed grade M

Total CPU time 24 secs
Total REAL time: 24 secs
Completely routed.
End of route.  892 routed (100.00%); 0 unrouted.
Checking DRC ...
No errors found.

Hold time timing score: 0, hold timing errors: 0

Timing score: 0

Dumping design to file attributes_attributes.dir/5_1.ncd.

All signals are completely routed.

PAR_SUMMARY::Run status = completed
PAR_SUMMARY::Number of unrouted conns = 0
PAR_SUMMARY::Worst  slack> = 0.531
PAR_SUMMARY::Timing score> = 0.000
PAR_SUMMARY::Worst  slack> = 0.235
PAR_SUMMARY::Timing score> = 0.000

Total CPU  time to completion: 24 secs
Total REAL time to completion: 25 secs

par done!

Copyright (c) 1991-1994 by NeoCAD Inc. All rights reserved.
Copyright (c) 1995 AT&T Corp.   All rights reserved.
Copyright (c) 1995-2001 Lucent Technologies Inc.  All rights
reserved.
Copyright (c) 2001 Agere Systems   All rights reserved.
Copyright (c) 2002-2013 Lattice Semiconductor Corporation,  All
rights reserved.
```

PAD Specification File

The PAD specification (.pad) file is an ASCII report file that lists all Programmable I/O Cells (PICs) used in the design and their associated primary pins. The PICs are listed by port name and by pin names.

To view the PAD Specification Report File in the Diamond environment:

- In the Design Summary pane of the Reports window, select **Signal/Pad**. Your report will appear in the pane to the right.
- Alternatively, use a text editor to open the .pad ASCII file from your project directory.

The .pad file includes the following information:

- Pinout by Port Name – This section lists the port names with primary pin designations. It also shows the buffer type and any associated attributes. Buffer type indicates the PIO mode. For example, a PIO in LVDS or LVPECL mode needs two bonded pads for differential signals that are both included in the .pad file.

  In most designs, the top-level HDL or schematic design will not include both true and complement sides of differential port signals. The designer assigns I/Os to pads (via the LOCATE preference) using a single signal, assuming that it will represent the true (+) signal. Based on an IOBUF preference type of LVDS or LVPECL, Diamond will automatically infer the complement (-) signal and assign it to the appropriate site on the device package.

  Port signals are reported using the syntax: <port name>+ for positive differential and <port name>- for negative differential in the Pinout by Pin Number table. If the source native generic database (.ngd) file does include LVDS or LVPECL type buffers where both true and complement ports exist, then the original signal names without +/- notation are used in the report.
- Vccio by Bank – This section lists the voltage by bank.
- Vref by Bank – This section shows the name and location of on-chip voltage references that have been set and the associated port groups or signals.
- Pinout by Pin Number – This section lists pin numbers, which includes primary pin number, the component name or reference voltage type, the buffer type, and any preferences assigned to the component. The Dual Function column identifies pins that can be used for I/O assignments or for another function, such as Vref. In the Pin info column, the status of any "unused" pin is reported as "unused, PULL:UP," because these pins will have an internal pull-up.
- Locate Preferences – This section lists LOCATE preferences that are in the .prf file. The list includes user-defined LOCATE preferences that were assigned during design entry and those that were set automatically by Place & Route.

Delay Statistics Report File

When you use the -y option from the command line or the “Create Delay Statistics File” option in the Strategies dialog box, each PAR run will generate a delay file (.dly). This file contains delay information for each net in the design. It includes:

- A listing of the 20 nets with the longest delays.

  An "e" preceding a maximum delay indicates that the delay shown is only approximate.

- A delay analysis for each net, including the net name, followed by the driver pin and the load pin(s).

The “Create Delay File” or -y option also appends a Delay Summary Report to the end of the Place & Route report. The first column of this summary gives the actual averages for the design. The figures in the second column, which are enclosed by parentheses, indicate a "worst case" scenario for the delay.

See Also “Running Place & Route Design” on page 612

- “Place & Route Design Options” on page 1144
- “Cost-Based Place & Route” on page 630
- “Timing-Driven Place & Route” on page 631
- “Place & Route Considerations” on page 634
- “Running PAR from the Command Line” on page 2505

Cost-Based Place & Route

The standard PAR package is a cost-based tool. This means that placement and routing are performed using various cost tables that assign weighted values to relevant factors such as constraints, length of connection, and available routing resources.

Placement The PAR process places the mapped physical design (.ncd file) in two stages: a constructive placement and an optimizing placement. PAR writes the physical design after the completion of each of these two stages.

During constructive placement, PAR places components into sites based on factors such as the following:

- Constraints specified in the input file (for example, certain components must be in certain locations)
- Length of connections
- Available routing resources
- Cost tables that assign random weighted values to each of the relevant factors. There are 100 possible cost tables.

Constructive placement continues until all components are placed. Optimizing placement is a fine-tuning of the results of the constructive placement.

Routing Routing also is done in two stages: iterative routing and delay reduction routing, which is also called cleanup. PAR writes the physical design (.ncd) only after iterations where the routing score has improved.

During iterative routing, the router performs an iterative procedure to converge on a solution that routes the design to completion or minimizes the number of unrouted nets.

During reduction routing, the router takes the result of iterative routing and reroutes some connections to minimize the signal delays within the device. There are two types of reduction (cleanup) routing that you can perform:

- A faster cost-based cleanup routing. This type of routing makes decisions by assigning weighted values to factors, such as the type of routing resources used, that affect delay times between sources and loads.
- A more intensive delay-based cleanup routing. This type of routing makes decisions based on computed delay times between sources and loads on the routed nets.

If PAR finds timing preferences in the preference file, timing-driven placement and routing is automatically invoked.

See Also ▶"Place and Route" on page 602

- "Timing-Driven Place & Route" on page 631

Timing-Driven Place & Route

In addition to cost-based PAR, you can run timing-driven placement and routing using the timing analysis engine.

This means that placement and routing is executed according to timing constraints (preferences) that you specify up front in the design process. The timing analysis engine interacts with PAR to ensure that the timing preferences you impose on the design are met.

To use timing-driven PAR, you simply write your timing preferences into a logical preference (.lpf) file in the design entry stage. The mapping process writes these preferences to the physical preference file (.prf), which serves as input to the Timing Wizard. If PAR finds timing preferences in the preference file, timing-driven placement and routing are automatically invoked.

Below is a brief description of timing preferences that PAR supports. For additional information, see "Applying Design Constraints" on page 360.

FREQUENCY Identifies the minimum operating frequency for all sequential input pins clocked by a specified net.

PERIOD Specifies a maximum clock period for all sequential input pins clocked by a specified net.

MAXDELAY Identifies a maximum total delay for a circuit net, path, path class, or bus in the design. Also specifies a maximum delay from a starting point or to an endpoint.

MAXSKEW Identifies a signal skew between the loads on a specified signal.

BLOCK Blocks timing checks on path classes, nets, buses, or component pins that are irrelevant to the timing of the design.

DEFINE STARTPOINT Identifies a starting point for the MAXDELAY FROM/TO preference (see MAXDELAY).

DEFINE ENDPOINT Identifies an ending point for the MAXDELAY FROM/TO preference (see MAXDELAY).

DEFINE PATH Specifies a path from a source component to a destination component.

DEFINE BUS Specifies a grouping of nets.

MULTICYCLE Allows for relaxation of previously defined PERIOD or FREQUENCY constraints on a path.

Each timing preference can generate many timing constraints for the design. For example, a PERIOD or FREQUENCY preference generates a timing constraint for every data path clocked by a specified net. A MAXDELAY for a designated path class generates a timing constraint for every path in the path class. Depending upon the types of timing preferences specified and the values assigned to the preferences, PAR run time might be increased.

When PAR is complete, you can verify that the design's timing characteristics (relative to the preference file) have been met by running TRACE (Timing Reporter and Circuit Evaluator), Diamond's timing verification and reporting utility. TRACE issues a report showing any timing warnings and errors and other information relevant to the design.

See Also

- "Applying Design Constraints" on page 360
- "Setting Timing Preferences" on page 492
- "Setting Preferences" on page 459
- "Preferences to Improve Timing" on page 380
- "Place and Route" on page 602
- "Cost-Based Place & Route" on page 630

Improving PAR Results for Highly Congested Routing

Some FPGA designs can exhibit a high degree of routing congestion after placement and routing. This can be due to design logic with many high fanout signals and a high degree of signal interconnect. Large state machines and control logic are common types of designs that lead to routing congestion.

Congestion can lead to large route delay or even prevent the routing phase of place and route from completing. If route delay represents a large fraction (>80%) of overall delay, the design might be a good candidate for constraint-driven and negotiation-based router options.

The default router is the negotiation-based router (NBR). The NBR involves an iterative routing that routes connections to achieve minimum delay cost. It does so by computing the demand on each routing resource and applying cost values per node. It will finish when an optimal solution is arrived at or the number of iterations is reached.

Congestion-driven PAR options for placement (CDP) and routing (CDR) can improve performance for a design that has multiple congestion hot-spots. The congestion display in Floorplan View or Physical View can help you visualize the routing congestion. Large congested areas might prevent congestion-based PAR options from finding a successful solution.

A combination of NBR, CDP, and CDR options can yield improved results.

Relative PAR run time and quality of results by option:

- Very congested, slower run time: NBR + CDP
- Moderate congestion, faster run time: CDR + CDP

To access options in Diamond for highly congested routing:

1. Choose **Project > Active Strategy > Place and Route Design Settings**.
2. In the Strategies dialog box, select **Routing method**.
3. Double-click the Value cell and choose a routing method—**NBR** or **CDR**.
4. In addition, set the following options based on the degree of congestion shown in your design:
   - Congestion-Driven Placement – 1 (on) enables congestion-driven placement under any circumstance. The Auto option automatically turns on congestion-driven placement when it is needed.
   - Congestion-Driven Routing – 1 (on) enables congestion-driven routing under any circumstance. The Auto option automatically turns on congestion-driven routing when it is needed.
5. Click **OK**.

These options are also available from the command line, as follows:

-exp parCDP (congestion-driven placement)

-exp parCDR (congestion-driven routing)

-exp parUseNBR (negotiation-based routing)

See Also ▶PAR Explorer options

▶ “Run Time Reduction” on page 609

Place & Route Considerations

Each Lattice device provides a clock distribution network that includes primary, secondary, and edge clocks for distributing high-performance clocks.

Primary clocks are automatically selected by the place and route process according to the load number and driver type. You can also define a signal as a primary clock using the USE PRIMARY preference. For details on the number of primary clocks available in your device target, refer to the device data sheet.

In primary clock placement, if the primary clock is a PIO, the placer will automatically place it into a "sweet site," a site that can be routed easily to the center. However, if the primary clock is driven by a PFU (internally generated primary clock), the placer will attempt to place it on a sweet site or a proximal location. If you locate a primary clock driver on a non-sweet site, the placer will issue a warning. During pre-placement, all PIO constraints are observed, which means that the pre-placement is always legal.

The placer prints out a list of selected primary clocks before the constructive placement. You should check that the primary clocks are selected properly.

See Also ▶“USE PRIMARY” on page 1268

Reducing Run Time in PAR

For all Lattice FPGA devices, the placer and router algorithms in Diamond have been optimized to improve run time in PAR. You do not have to set any options to benefit from these improvements. Additionally, the default negotiation-based router (NBR) helps reduce run time on designs that are difficult to route.

NBR uses a negotiation technique to arrive at viable routing solutions. At first, the NBR Router works to establish a minimum delay cost of each connection, allowing wiring shorts and assuming all nodes are available for use. Using this as a basis, the NBR Router algorithm implements an iterative process that places demand value on node resources, using a cost-based system. During this iterative process, signals that used nodes in congested areas are reprioritized to less expensive areas of congestion and shorts are resolved. The tool “negotiates” the expense of each connection based on this cost system until it reaches the most effective routing solution for the design.

For a moderately congested design, you might want to switch to the congestion-driven routing method (CDR). The NBR and CDR Routing Method options are available in the Place & Route Design settings of the Strategy

dialog box. You can also set the routing method from the command line using the PARUSENBR and PARCDR options.

See Also ▶PAR Explorer options

- "Hold-Time Error Correction" on page 605
- "Run Time Reduction" on page 609

Running Multiple PAR Jobs in Parallel

The PAR Multi-tasking option allows you to use multiple machines (nodes) to run multiple place-and-route jobs at the same time instead of serially. You can run this feature in the Diamond environment. Use the "Multi-tasking Node List" option in the Place and Route Settings of the Strategies dialog box, or use the PAR-m option from the command line.

This ability to run multiple jobs on different nodes simultaneously can significantly reduce the time it takes to complete these runs. Otherwise, it would take the cumulative time for each job to complete by itself in a consecutive manner, and this could mean the difference between 1 hour and 10 hours of total time.

The PAR multi-tasking option is supported three ways:

- Local multi-tasking, where multiple PAR runs are executed on a single machine.
- Networked multi-tasking, where multiple PAR runs are executed on multiple machines.
- A combination of local and networked.

Creating a Node List File For a PC or for UNIX/Linux, you must first create an ASCII-based node list file input that specifies the candidate machines that will allow multiple PAR jobs to be run in parallel.

The node list file contains node description blocks of information on separate lines for each machine in the format shown below:

[<*node_name*>]

System = <*linux* | *pc*>

Corenum = <*number_of_cores*>

Env = <*file_name*>

Workdir = <*directory_name*>

[<*node_name2*>]

System = <*linux* | *pc*>

Corenum = <*number_of_cores*>

Env = <*file_name*>

Workdir = <*directory_name*>

where:

The [<*node_name*>] value contains the machine name in square brackets. For example, if your machine name is *jsmith*, this line would simply show [jsmith] on the first line.

Note

For UNIX/Linux, you can use many networked nodes to run the PAR job, so the node list will have a corresponding number of node description blocks in it. For PC, there will only be one node description block in the node list since you can only employ one PC that has multiple cores. This feature does not currently support a host of networked PCs.

The System node description line lists the platform the machine runs on. This line can only take Linux or PC as a value.

Corenum specifies how many CPU cores there are on the machine. Changing it to zero disables the node from running PAR tasks.

Env specifies an environment setup file to be run before the multiple PAR job is run on the remote machine. If the remote machine environment is ready, then this line can be omitted. To test if a remote machine is ready, run the following command to see if PAR can be run remotely:

ssh <*remote_machine*> <*par_cmd_options*>

Workdir specifies in which working directory on the remote machine the PAR multi-task job should be run. If account permissions allow you to get into that directory automatically after login, this line can also be omitted. You can test if a remote user account needs to set the working directory by running the following command and see if the design can be found:

ssh <*remote_machine*> ls <*ncd_file*>”

If the design is found after issuing the above command, there is no need to specify this line.

If Workdir is set to a location, it must be set to a directory where ssh into that nodename has read/write access in that directory.

General Node List Rules:

- You must use the equals sign for each node description line.
- Only the names of files and directories are case sensitive on UNIX and Linux. Node description line markers, for example, “Corenum **=**” can appear in upper, lower, or mixed case.

- The [<*node_name*>], Corenum and System lines are required for any node list file.
- The Env and Workdir node description lines are optional.
- For PC, the nodelist file should only include one node description block on the local machine. Remotely networked PCs are not supported. Any entries for remote nodes on a PC network will be discarded by the PAR multi-tasking feature.
- For UNIX/Linux, the nodelist file can have multiple machine nodes; but it should not include PC type nodes.
- Any text following double forward slashes, "//", will be considered as a comment and ignored, except when those double forward slashes are enclosed in double quotation marks. If you need to specify a path containing a double forward slash, you can do so by enclosing it in quotes. Comments can be added at any position in a line.
- Whitespace around node description line markers is ignored.
- Blank lines, including those with comments, are ignored.

PAR Multi-Tasking Environment Setup For PC, running PAR remotely is not supported, so multiple PAR processes can only run in parallel on one PC. Since the child PAR process takes the environment from the parent, no environment variable setup is necessary.

For UNIX/Linux, there are two ways to set up the environment:

- Use the account login profile and source files.

  For the account to allow PAR to be run remotely, it should have an .rc file (.bashrc if the bash shell used) that runs automatically when a secure shell (ssh**)** call is received. This script file needs to set up the environment to the correct Diamond path and “cd” to the directory where the NCD file exists.

  The following three environment variables have to be set properly for PAR to run: PATH, FOUNDRY, and LD_LIBRARY_PATH. To set the environment variables, see “Setting Up the Environment to Run Command Line” on page 2470. You can test if the remote environment is set up correctly by running the following command locally:

  ssh <*remote_machine*> par

  You will see the PAR help page if the environment setup is correct.

- Use Env and Workdir node description lines in the node list file.

  The PAR Multi-tasking option will know which script to run and in which working directory to run the par command after it reads the node list. The spawned task on the remote machine will run the “Env” file, then it will “cd” to the working directory before it runs PAR.

  Note

  The result of MAP (_map.ncd) must be in a directory location in which every ssh to every nodename has access.

There is also UNIX/Linux environment setup information for this feature in the Using the PAR Multi-Tasking (-m) Option section of the topic "Running PAR from the Command Line" on page 2505.

Running Multiple PAR Jobs in Parallel in Diamond When running the multi-tasking option on the PC, you should use a dual processor or multiple processor configuration, or the feature will provide no benefit. Currently, you cannot run the multi-tasking option using multiple networked PCs.

To run the PAR Multi-tasking on the PC with multiple processors:

1. Prepare a node list ASCII text file that identifies your machine (lpass4), system type (PC), and number of core processors (2). Include any desired comment lines.

   ```
   // This file contains a profile node listing for a multipar
   // PC job.
   [lpass4]
   SYSTEM = PC
   CORENUM = 2
   ```

   You must use the format above for the node list file and fill in all required parameters. Parameters are case insensitive. The node or machine names are given in square brackets on a single line.

   The System parameter can take the Solaris, Linux, or PC values, depending upon your platform. However, the PC value cannot be used with either Solaris or Linux because it is not possible to create a multiple computer farm with PCs. Corenum refers to the number of CPU cores or processors available on that single PC. Setting it to zero will disable the node from being used. Setting it to a greater number than the actual number of CPUs will cause PAR to run jobs on the same CPU, lengthening the run time. No further parameters are necessary on the PC.

2. Save your node list file and copy it to your top-level project directory. In our example, we named the node list my_nodelist.txt.
3. In Diamond, click **Project > Active Strategy > Place and Route Settings**.
4. Double click the **Value** cell for the **Multi-Tasking Node List** option and click the browse button.
5. Navigate to the file name in your project directory. If the file is located somewhere else, you must use an absolute file and path name, for example, *C:/my_projects/my_nodelist.txt*. PAR will issue an error message if it cannot find your node list file, for example:

   ```
   ERROR - par: Node name file  does not exist.
   ```

   If it cannot find your node list, PAR will continue in a serial fashion without using the node list file to specify the use of multiple processors.

6. Click **OK**.

   In our example, we opted to change the number of placement iterations to 3, the placement iteration starting point to 3, and limit the number of best saved runs to 2.

7. In the Process view in the Diamond environment, right click **Place & Route Design** and choose **Run** from the pop-up menu to run the flow through the Place & Route process.

   Rerun will run just the one process over again and nothing else. Rerun All will rerun the design flow up to the place and route process if selected.

   In the Output view or Automake Log, the multipar run will show that it ran all jobs and completed successfully.

```
---- Multipar Tool ----
Running par. Please wait . . .
Starting job 5_3 on node lpass4
Starting job 5_4 on node lpass4
Finished job 5_4 on node lpass4
Starting job 5_5 on node lpass4
Finished job 5_3 on node lpass4
Finished job 5_5 on node lpass4
Exiting par with exit code 0
Exiting multipar with exit code 0
Done: completed successfully.
```

   Notice that we chose the starting point of 3, so the jobs start at 5_3 instead of 5_1.

   After the run, PAR creates a <project_name>.dir directory in your top-level project directory with the respective NCD design files along with their .par and .pad report files. The following are example directory contents:

```
my_project.dir:
5_3.ncd         5_3.par         5_4.pad         test_view.par
5_3.pad         5_4.ncd         5_4.par
```

   In this example output directory only the best two runs were saved, as specified. Because the start point was the third iteration, it starts with 5_3. The 5 represents the effort level and the 3 the number of the run.

To learn more about how to use this feature from the command line, see the subsection Using the PAR Multi-Tasking (-m) Option of the topic “Running PAR from the Command Line” on page 2505. There is also information there about general usage, environment setup, screen output, interrupting jobs, and requirements.

See Also ▶“Running PAR from the Command Line” on page 2505

▶ “Using the PAR Multi-Tasking (-m) Option” on page 2519

Bit Generation

The **Bitstream File** process in the Process view or bit generation (**bitgen**) program takes a fully routed physical design, in the form of a circuit description (.ncd) file as input and produces a configuration bitstream (bit images). The bitstream file contains all of the configuration information from

the physical design that define the internal logic and interconnections of the FPGA, as well as device-specific information from other files associated with the target device.

The data in the bitstream can then be downloaded directly into the FPGA's memory cells or used to generate files for PROM programming. You can run **bitgen** from the Diamond window by double-clicking the **Bitstream File** process or from the command line.

See Also "Generating Bitstream Files" on page 643

- "Bit Generation Considerations" on page 644
- "Bit Generation Input Files" on page 642
- "Bit Generation Output Files" on page 642
- "Bit Generation Options" on page 640
- "Running Bit Generation from the Command Line" on page 2538

Bit Generation Options

Bit generation options provide you with control over the bit generation process. Bit generation options are accessed from the Strategy dialog box or the **bitgen** program from the command line. These options allow you to control the format of the bitstream output.

For more information on bit generation options see the following online help topics:

- Create Bit File, (ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M only) will create or prevent creation of a bitstream file.
- No Header, (ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M only) allows you to preclude the header information from the output bitstream.
- Output Format, (ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M only) specifies the type of bitstream to create.
- Run DRC, runs a physical design rule check and saves the output to the Bit Generation report (.bgn file).
- Search Path, (LatticeSC PCS and some Sysbus only) specifies a file path or paths you want **bitgen** to search for a auto-configuration file.

Advanced Options: Address, (LatticeSC/M) is the bitstream mode.

- Chain Mode, allows daisy chaining with other devices.
- Enable I/O in Re-configuration, (LatticeSC/M only) programs the FPGA to drive outputs (not be tristated) during configuration/reconfiguration.
- External Clock, (LatticeSC/M only) is used for testing Master configuration modes.

- Grant Timeout, (LatticeSC/M only) controls an internal system bus grant counter that counts the number of HCLK (sysbus clk) cycles that go by before the grant signal is taken away from the master.
- LengthBits, (LatticeSC/M only) allows you to specify the number of allowable bits in the length field.
- Output Zero Frames, (LatticeSC/M) controls whether zero frames are written to the output bitstream.
- Register Configuration, (LatticeSC/M only) resets the PFU registers before configuration.
- Reset Config RAM in Re-configuration, re-initializes the device (reset) when you download a bitstream by default. If No reset is selected, the current configuration is retained allowing for additional bitstream configuration.
- SPI Start Address, specifies SPI start address. The SPI start address is a 6 or 8 digit hex string prefixed with 0x. The default is 0x00000000.
- Startup Clock, (LatticeSC/M only) uses CCLK or a User clock during startup.
- Sysbus Clock Config, (LatticeSC/M only) sets the system bus clock to either reset or not to reset after reconfiguration.
- Sysbus Config, (LatticeSC/M only) sets the system bus to either reset or not to reset after reconfiguration.
- Wait State Timeout, (LatticeSC/M only) controls an internal system bus timeout counter.
- XRES Calibration Control, (LatticeSC/M only) controls the calibration circuit connected to XRES pin.

JEDEC File Options: These options are for a separate process that generates JEDEC standard files for MachXO, MachXO2, MachXO3D, MachXO3L, and LatticeXP/2 devices instead of those by the generated Bitstream File (bitgen) for the other FPGA device families.

- Chain Mode, allows daisy chaining with other devices.
- Disable UES, (MachXO, MachXO3D, MachXO2, MachXO3L, LatticeXP/2 only) will generate JEDEC file that will not contain User Electronic Signature Data.
- ReadBack, (LatticeXP/2 only) allows you to extract the configuration data stored in an FPGA in order to verify the configuration.
- ReadCapture, (LatticeXP/2 only) enables or disables readback of the configuration bitstream.
- Reset Config RAM in Re-configuration, re-initializes the device (reset) when you download a bitstream by default. If No reset is selected, the current configuration is retained allowing for additional bitstream configuration.
- Run DRC, runs a physical design rule check and saves the output to the Bit Generation report (.bgn file).

See Also "Bit Generation" on page 639

- "Bit Generation Input Files" on page 642
- "Bit Generation Output Files" on page 642

Bit Generation Input Files

The following files are input to the **Bitstream File** process or the **bitgen** program:

- **Native Circuit Description File** (.ncd) — Input file for bit generation is an active, fully routed NCD file. This file is a physical description of the design in terms of the components in the target device.

See Also

- "Bit Generation" on page 639
- "Bit Generation Output Files" on page 642

Bit Generation Output Files

The following files are possible output to the **Bitstream File** process or the **bitgen** program:

- **Bit File** (binary) — binary (.bit) bitstream. Binary bitstream files are the default output of the bitstream process and contain the configuration information in bitstream (zeroes and ones) that is represented in the physical design (.ncd) file.
- **Raw Bit File** (ASCII) — ASCII (.rbt) bitstream. The Raw Bit File is a text file containing ASCII ones and zeros representing the bits in the bitstream file. If you are using a microprocessor to configure a single FPGA, you can include the Raw Bit file in the source code as a text file to represent the configuration data. The sequence of characters in the Raw Bit file is the same as the bit sequence that will be written into the FPGA. The .rbt file differs from the .bit file in that it contains design information in the first six lines.
- **Mask File** — mask (.msk) bitstream. Used to compare relevant bit locations for executing a read back of configuration data contained in an operating FPGA.
- **Bit Generation Report File** — bit generation report (.bgn). Outputs information on a bit generation (**bitgen**) run and displays information on options that are set. This file is output by default and will be given the name, design_name>.bgn.

See Also

- "Bit Generation" on page 639
- "Bit Generation Input Files" on page 642

Generating Bitstream Files

In Lattice Diamond, bitstream generation is automatically performed when you run the **Bitstream File** process for a a full design flow or a partial design flow. You can set bit generation options before running a design flow by setting a strategy.

To generate bitstream files from Diamond:

1. In the Diamond File List view, double-click the target strategy.

   The Strategy dialog box opens.

2. Under Process, select **Bitstream**.
3. In the right-hand pane, double-click the Value box for the Bitstream option that you want to edit, enter the new value or select a value from the drop-down list, and click **Apply**.

   Note

   When you highlight an option, its brief definition is displayed at the bottom of the dialog box. You can also click **F1** to view the detailed description.

4. When you finish, click **OK** to close the Strategies dialog box.
5. In the Diamond Process view, double-click **Bitstream File** to generate the bitstream files..

   Note

   MachXO3LF designs that do not use User Flash Memory (UFM) can be migrated to lower-cost MachXO3L devices, and a valid bitstream can be generated. The following two options are available when MachXO3LF is the selected device:

   - JEDEC File (XO3L Migration)
   - Bitstream File (XO3L Migration)

   This migration path is a time-saving feature that allows users to validate functionality and timing on a MachXO3LF project without having to recompile the design for the MachXO3L device. MachXO3L devices also do not support soft error correction or security encryption.

See Also ▶"Bit Generation" on page 639

- "Bit Generation Input Files" on page 642
- "Bit Generation Output Files" on page 642
- "Bit Generation Considerations" on page 644
- "Running Bit Generation from the Command Line" on page 2538

JTAG Setup

Using the **bitgen** program from the command line, you can generate setup bitstreams (.jbt) files to set JTAG port read and write for the FPGA device using the -J option. Downloading these setup bitstreams requires the serial cable with a JTAG cable connector.

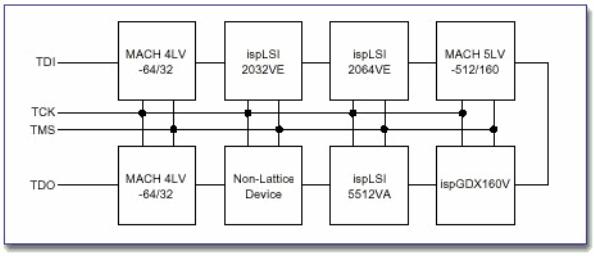
See Also ▶"Running Bit Generation from the Command Line" on page 2538

JTAG Scan Chains

A scan chain can include any programmable or non-programmable device compliant with IEEE-1149.1. It can also include any programmable devices that are compatible with IEEE-1149.1 but do not have a boundary scan register. This decision should be made on the basis of the test methodology employed for the board. If the test methodology employed is the traditional bed-of-nails approach used on board test systems, all the devices can be included in the same chain.

All scan chains use the simple four-wire TAP. The TCK and TMS pins are common to all devices included in the chain. TDI and TDO are daisy-chained from one device to the next. The input to the chain is TDI, and the output from the chain is TDO. A diagram demonstrating a simple scan chain is shown below.

Figure 111:



Bit Generation Considerations

Note the following user information that may require additional steps to implement bit generation options properly.

- If the device is to be configured in other than a serial (master or slave) mode, there are some limitations involving startup options and configuration pins used as outputs after configuration.
- Some former **-g** options for **bitgen**, which correspond to former strategy settings for **Bitgen** in the Strategies dialog box, are now handled by setting SYSCONFIG preference keyword values. See the preference "SYSCONFIG" on page 1250 topic for details and usage.

See Also ▶"Bit Generation Options" on page 640

Analyzing your Design using Soft Error Injection

Ambient radiation occasionally collides with a circuit component in a way that alters its logical state. This effect is strongest at high altitudes, where the Earth's protective electromagnetic field is weaker. SEI (Soft Error Injection) Editor allows you to generate single-bit errors, insert them into a bitstream, and detect them for analysis, simulating the effect of radiation damage on the device's configuration memory. SEI Editor currently supports the following devices:

- ECP5U
- ECP5UM
- MachXO2
- MachXO3D
- MachXO3L/LF

SEI Editor is available after you have placed and routed your design and generated a bitstream.

See Also "Running SEI Editor" on page 645

- "Adding and Configuring a New Error with SEI Editor" on page 647
- "Removing an Error with SEI Editor" on page 647
- "Inserting an Error into a Device Using SEI Editor" on page 647

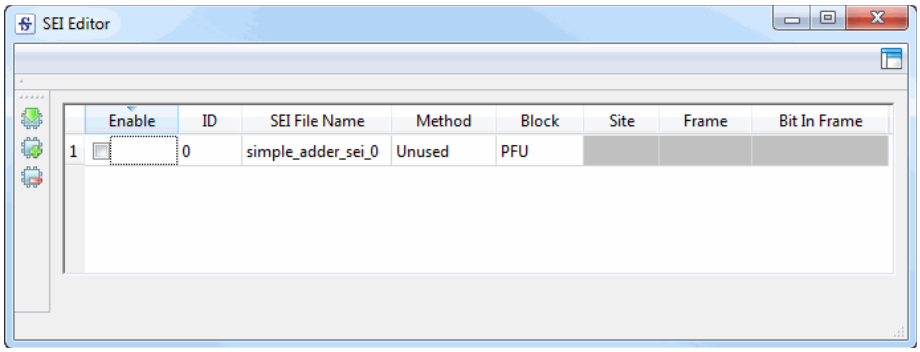
Running SEI Editor

You can run SEI Editor after you have placed and routed your design and generated a bitstream.

To run SEI Editor:

- Choose **Tools > SEI Editor**

  SEI Editor appears.



SEI Editor can manage multiple errors with a variety of options. The SEI Editor toolbar has three buttons:

- **Run** — Run all selected error insertions.
- **Add** — Add a new error for insertion.
- **Remove** — Remove the selected error insertion.

SEI Editor has the following columns:

- **Enable** — Select this option to insert the error described on the current row.
- **ID** — The error's ID number. This value is read-only.
- **SEI File name** — The name of the file containing the inserted error. You can modify this filename by double-clicking it and entering a new name.
- **Method** — This option determines whether the error is inserted into an unused or random block. Selecting Unused inserts the error into an unused block, which will not affect the design logic. Selecting Random inserts the error into a random block, which may or may not affect the design logic. You can change an error's Method by double-clicking it and selecting a new value from the drop-down menu.
- **Block** — The block where the error was inserted, in one of the following categories:
  - PFU
  - EBR
  - DSP
  - ANY
  - Routing
  - Unclassified

  You can direct errors into unused PFU, EBR, or DSP blocks, or into a randomly-chosen unused block. You can change the destination Block by double-clicking it and selecting a new value from the drop-down menu.
- **Site** — The exact Site of the inserted error. Errors inserted into a Routing block or Unclassified block do not have a Site. This field is read-only.
- **Frame** — The data frame of the inserted error. This field is read-only.
- **Bit In Frame** — The bit of the inserted error within the displayed data frame.

See Also
- "Analyzing your Design using Soft Error Injection" on page 645
- "Adding and Configuring a New Error with SEI Editor" on page 647
- "Removing an Error with SEI Editor" on page 647
- "Inserting an Error into a Device Using SEI Editor" on page 647

Adding and Configuring a New Error with SEI Editor

You can use SEI Editor to add new errors to your design.

To add and configure a new error:

1. Click **Add** .

   The tool adds a new, unconfigured error to the list.
2. Select the option in the Enable column to enable the new error.
3. Choose a new filename for the inserted error by double-clicking the current filename, typing a new one, and pressing Enter.
4. Choose a method by double-clicking the current method and selecting your preferred method from the drop-down menu.
   - If this error's method is Unused, you can choose a block by double-clicking the current block and selecting your preferred block from the drop-down menu.

See Also "Analyzing your Design using Soft Error Injection" on page 645

- "Running SEI Editor" on page 645
- "Removing an Error with SEI Editor" on page 647
- "Inserting an Error into a Device Using SEI Editor" on page 647

Removing an Error with SEI Editor

You can remove a previously inserted error from your design using SEI Editor.

To remove an error:

1. Click a cell in the row of the error you want to remove.
2. Click **Remove** .

See Also "Analyzing your Design using Soft Error Injection" on page 645

- "Running SEI Editor" on page 645
- "Adding and Configuring a New Error with SEI Editor" on page 647
- "Inserting an Error into a Device Using SEI Editor" on page 647

Inserting an Error into a Device Using SEI Editor

Once you have inserted an error into your design, you can insert the error into your device using Diamond Programmer.

To insert the selected errors into a device:

1. Click **Run** .

   The SEI tool generates a partial bitstream with the selected errors.

2. Program the original bitstream into the device using Diamond Programmer. Refer to "Programming the FPGA" on page 802.
3. In Diamond Programmer, program the partial bitstream into the device using the **Static RAM Cell Background Mode** and **XSRAM SEI Fast Program** Programmer settings**.**

See Also "Analyzing your Design using Soft Error Injection" on page 645

- "Running SEI Editor" on page 645
- "Adding and Configuring a New Error with SEI Editor" on page 647
- "Removing an Error with SEI Editor" on page 647

PROM Generation

After creating a bitstream file through bit generation, you can store the configuration information in a PROM file. The **PROM File** process takes one or more bitstreams (.bit or .rbt files) and creates a PROM-formatted file in one of three widely used PROM formats: Intel's MCS-86, Motorola's EXORMACS, or Tektronix's TEKHEX.

There are three common storage methods:

- One design in one PROM.
- Individual designs stored separately in one PROM. This is useful when you want to use a single PROM to reprogram the same device with different designs.
- Several different designs concatenated in the same or multiple PROMs. This approach is likely when you want to program several devices in a daisy chain or when using a PROM that is too small to store the complete bitstream for the target FPGA.

See Also "Generating PROM Files" on page 649

- "PROM Generation Input Files" on page 649
- "PROM Generation Output Files" on page 649

PROM Generation Options

PROM generation options provide you with control over the PROM generation process. PROM generation options are accessed from Properties of the **PROM File** process. These options allow you to control the format of the PROM file output. PROM.

For more information on PROM generation options see the topic "PROM Data Output Format" on page 1158.

See Also "PROM Generation" on page 648

- "PROM Generation Input Files" on page 649

▶ “PROM Generation Output Files” on page 649

PROM Generation Input Files

The following files are possible input to the **PROM File** process:

- **Bit File** (binary) — binary (.bit) bitstream. Binary bitstream files are the default output of the bitstream process and contain the configuration information in bitstream (zeroes and ones) that is represented in the physical Native Circuit Description (.ncd) file.
- **Raw Bit File** (ASCII) — ASCII (.rbt) bitstream. The Raw Bit File is a text file containing ASCII ones and zeros representing the bits in the bitstream file.

The input to the Prom Generation process is one or more bitstreams (.bit or .rbt files). These files are generated in the **Bitstream File** process during a design flow run in Diamond or by using the Bit Generation (**bitgen**) command from the command line.

See Also ▶“PROM Generation” on page 648

▶ “PROM Generation Output Files” on page 649

PROM Generation Output Files

The following files are possible output to the **PROM File** process or the **promgen** program:

- **PROM file** — contains configuration information used to program the PROM. This file will have an extension of .mcs (Intel MCS-86), .exo (Exormacs), or .tek (TekHex), depending upon the selected PROM format.
- **PROM Image File** — contains a memory map of the newly created PROM file showing the starting and ending PROM address for each bitstream (.bit file) loaded. This file has a .prm extension.

See Also ▶“PROM Generation” on page 648

▶ “PROM Generation Output Files” on page 649

Generating PROM Files

After creating a bitstream file with bit generation, you may want to store the configuration information in a PROM. You can use the bitstream file you have created by running the design flow as your input for the PROM Generation. When using the Diamond window, the bitstream file will be in the project directory.

PROM options should be set before running the PROM generator. The PROM generator takes bitstreams and creates PROM image files for PROM configuration. You can set PROM options in the Strategy dialog box.

To generate PROM files from the Diamond window:

1. In the Diamond File list view, double-click the target strategy.

   The Strategies dialog box opens.

2. Under Process, select **Bitstream**.
3. In the right-hand pane, double-click the Value box for the Bitstream setting that you want to edit, enter or select the new value, and click **Apply**.
4. When you finish, close the Strategies dialog box.
5. In the Diamond Process view, double-click the **PROM File** process to generate the PROM files.

See Also

- "PROM Generation" on page 648
- "PROM Generation Input Files" on page 649
- "PROM Generation Output Files" on page 649

Chapter 8

Using Incremental Design Flow

The Incremental Design flow feature is supported for LatticeECP2M and LatticeECP3 devices. Incremental Design is a design methodology that preserves certain process results and performance on portions of a design while reducing re-processing time by focusing on other parts of the design during multiple design processing iterations. It accomplishes this by using design partitions you define in synthesis. Partitions allow you to lock down and preserve timing stability by preventing specified modules from being changed in the next design iteration.

In addition to partitions, the Incremental Design flow uses a previous design file to guide the rest of your design implementation toward smaller, more incremental changes. Using this strategy, you no longer have to re-implement your design from the beginning which forces changes that effect the technology mapping and layout of your entire design. Using a reference design, your previous I/O placement also remains intact.

You might consider using the Incremental Design flow for the following reasons:

- It reduces re-processing runtime during multiple iterations.
- It preserves previous timing results on portions of a design during multiple iterations.
- It minimizes runtime variations for design changes during multiple iterations.
- It helps achieve timing closure by partition planning and coding improvement.

The Incremental Design flow in Diamond requires you to first specify design partitions during synthesis in the top-level Synplify FPGA Design Constraints (.fdc) file. Design partitions are defined by the “Compile Points” feature in Synplify Pro, which enables you to create constraints in the .fdc file that tell the synthesis compiler to preserve and timestamp modules. So, partitions allow you to “lock” these portions of your design to preserve the logic through synthesis and mapping if there is no source code change to a partition. Place and Route (PAR) later will determine how components fall into the design layout with the aid of a partition planning utility within the PAR tool that is specific to this flow in Diamond.

Once you associate your .fdc file with your Diamond project and turn on the incremental flow mode, the PAR output of this mode is a reference physical

design (NCD) file. Diamond incremental flow will use this placed-and-routed reference NCD to help guide your design through the next processing iteration and take advantage of improved runtime and timing stability.

Through appropriate partition planning and coding improvement, the incremental design methodology will help achieve timing closure through multiple iterations. With today's ever-increasing FPGA densities to meet more complex applications, it can be challenging to meet your design goal. These designs are usually comprised of multiple function blocks and each block may have significant amount of logic or it may perform a specific task that requires being addressed separately using "divide and conquer" approach.

Incremental Design allows you to isolate and lock down satisfactory portions of such a design and focus on others. At a minimum, reducing your runtime on larger designs using this methodology will give you more time to work on portions of your designs that need more work.

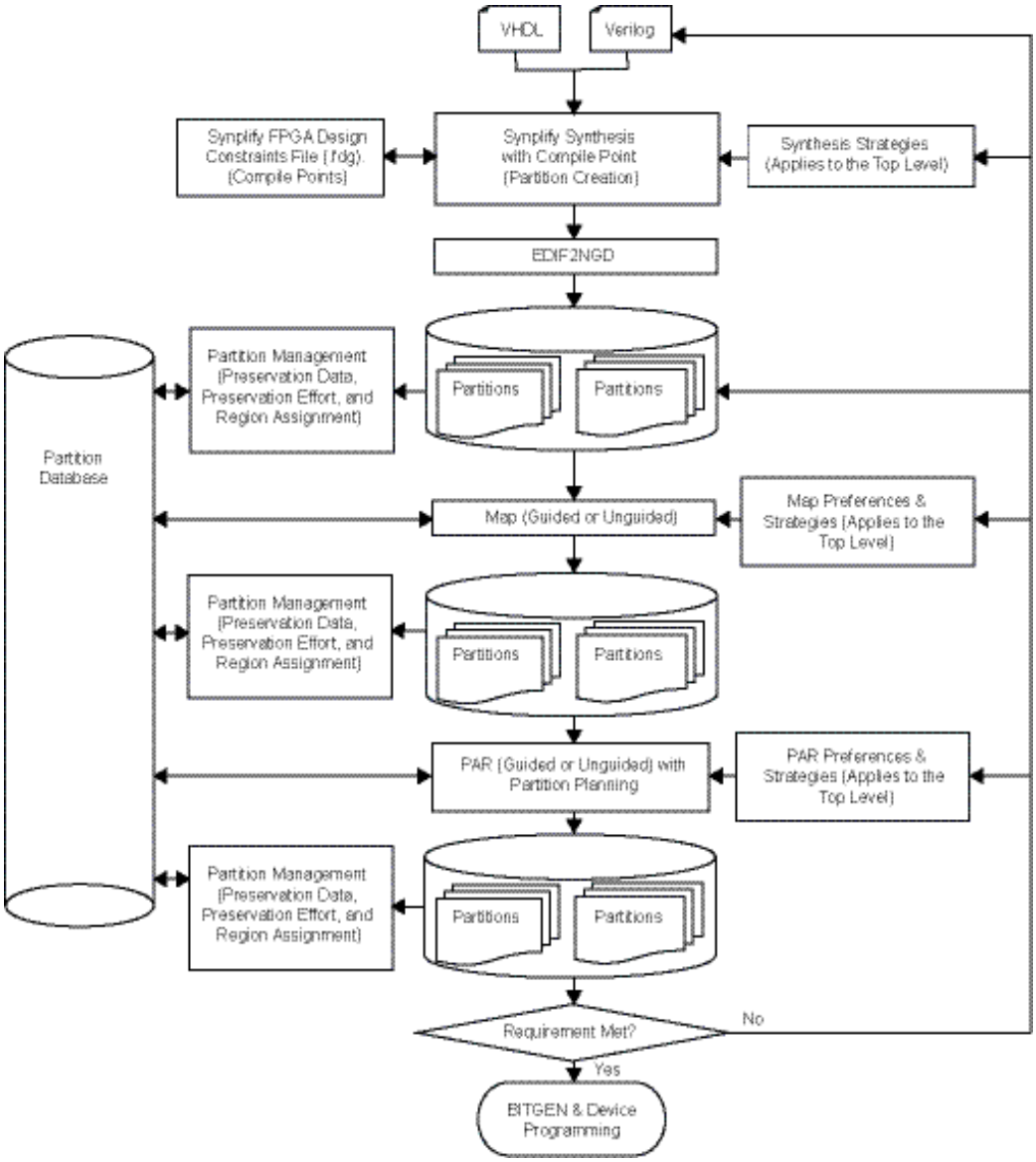
The Incremental Design Data Flow The section describes each design entry and implementation phase of the Incremental Design flow in terms of how data is processed in the Diamond environment through its various core tools. Figure 1 below illustrates the flow to provide you with a good basic understanding of how the flow works within the system.

The following list provides a high-level outline on how the Diamond Incremental Design flow works and how data is processed through each step:

1. You begin by defining your design partitions using "compile points" in top-level .fdc file using Synplify Pro. The type of compile point that is required for Incremental Design flow in Synplify Pro is {locked, partition}.
2. Synplify Pro runs a partition-based synthesis using compile point constraints. The output EDIF file has "partition" properties labeled on modules instances. These include timestamps from Synplify Pro.
3. The Design Translation process (EDIF2NGD and NGDBUILD) recognizes and honors "partition" properties and passes them onto their output NGO and NGD files, respectively. Other than passing the properties, these two processes are exactly the same as in the normal flow.
4. The Map process can optionally use the previous PAR NCD result as the reference to guide the process for each partition, according to user settings set through the Partition Manager. Otherwise, new mapped data for a partition (or the whole design, depending on user settings) will be generated.
5. The PAR process can optionally use the previous PAR NCD result as the reference to guide the process for each partition, according to user settings set through the Partition Manager. Otherwise, new PAR data for a partition (or the whole design, depending on user settings) will be generated.
6. View process reports in the Diamond Report window. If the PAR process finished successfully (err code 0, regardless of timing score), its result is ready to be used as the reference for the next iteration.

This guide consists of the following sections.

Figure 1: Incremental Design Data Flow



"Running the Incremental Design Flow" on page 655

This section focuses on running the actual Incremental Design flow in Synplify-Pro and Lattice Diamond.

"Using Partition Manager" on page 664

This section describes the Diamond Partition Manager tool in detail and its role in the flow.

"Recommended Strategies for Incremental Design" on page 677

This section provides advice to make the best strategic design decisions to help you avoid issues during the flow and get the best results.

"Troubleshooting Error and Warning Messages" on page 685

This section lists any common errors or warnings that you might encounter in the flow and provides you solutions for working around them.

Running the Incremental Design Flow

This section provides all description and procedural information you need to know to run a complete Incremental Design flow through Diamond software. A list of Tcl commands is provided in “Incremental Design Flow Tcl Commands” on page 2633.

Along with the steps provided within this chapter, there are two additional requirements for running the Incremental Design flow in Diamond. These requirements are as follows:

- Create a Synopsys FPGA Design Constraints (.fdc) file with compile points in Synplify Pro so that the Incremental Design flow can function properly within Diamond.
- All work in your Diamond project directory should be kept within one Diamond project implementation so that your reference files do not become disassociated with your target design.

Take the following steps to implement an incremental design in Diamond:

1. Create a new Diamond project and implementation; add HDL files as necessary.
2. Turn on incremental flow by choosing **Design > Enable Incremental Design Flow.**
3. Launch the Lattice OEM version of Synplify Pro from within Diamond. The Synplify Pro project should be created automatically for you. In Synplify Pro, create a top-level Synopsys FPGA Design Constraints (.fdc) file. This file is necessary for defining design partitions for synthesis. See “Creating Partitions in Synplify Pro” on page 656.
4. In Diamond, add your .fdc file to incremental flow enabled implementation. See “Running Incremental Design in Diamond” on page 658.
5. Run synthesis, Map, and PAR in Diamond. You can run this from the Tcl Console if you prefer to do so. See “Running Incremental Design in Diamond” on page 658.
6. Analyze static timing using the Timing Analysis view (Timing Preference file) or the Place & Route Trace process (TRACE Report file) to determine how close you are to meeting your design’s timing constraints.

   The TRACE report and Timing Analysis view analysis tools should provide indicators that tell you about resource utilization and alert you to any possible timing issues. You should also review Map and PAR reports. Review these reports to ensure that the data results make sense in regard to what you would expect to see in your design.

   If you are not already aware of the module or modules that you need to re-implement without locking down its logic using synthesis directives for recompile and re-synthesis, the reports should make this clearer. See “Viewing Reports for Incremental Design” on page 662.
7. if PAR finishes successfully, the PAR NCD will be used as a reference automatically during the next processing iteration. If necessary, you can modify your code; change strategies, preferences or partition management settings; and run the next processing iteration

To perform these steps follow the detailed procedures described in “Creating Partitions in Synplify Pro” on page 656 and “Running Incremental Design in Diamond” on page 658, respectively.

Creating Partitions in Synplify Pro

You create design partitions for your source files by creating a top-level Synopsys FPGA Design Constraints (.fdc) file in Synplify Pro. You need to generate an .fdc file and later associate it with your Diamond project for the Incremental Design flow to work properly and generate data later in the flow.

To create an .fdc file using the standalone Synplify Pro tool:

1. Open your incremental design flow project in Diamond. This procedure assumes that you have already created a project and an implementation, turned on the incremental flow for the implementation for your design (see “Running Incremental Design in Diamond” on page 658), and imported your source files into your Diamond project.
2. In Diamond, choose **Tools > Synplify Pro for Lattice** or click the  toolbar button. Synplify Pro opens.

   Notice that Synplify Pro automatically imports in all of your source HDL files and uses your project name in Diamond but appends “_syn” to the project name. In addition, it sets the same device family in both implementation and synthesis projects. It is important to open Synplify Pro from Diamond instead of launching from the Start menu or from some other installation so that the synthesis project inherits the project characteristics from Diamond.

   Now you will see your source files in a folder in your Project view. For example, if they are Verilog files the folder will be named accordingly.
3. Choose **Run > Compile Only** or **F7**. This enables the Synopsys FPGA Design Constraints (.fdc) file to be initialized, which allows you to define compile points.
4. Click the  **New Constraint File** toolbar icon.
5. In the New Constraint File dialog box, click **Constraint File (Scope)** and, in the Create a New Scope File dialog, ensure that the default options are checked or selected and click **OK**:

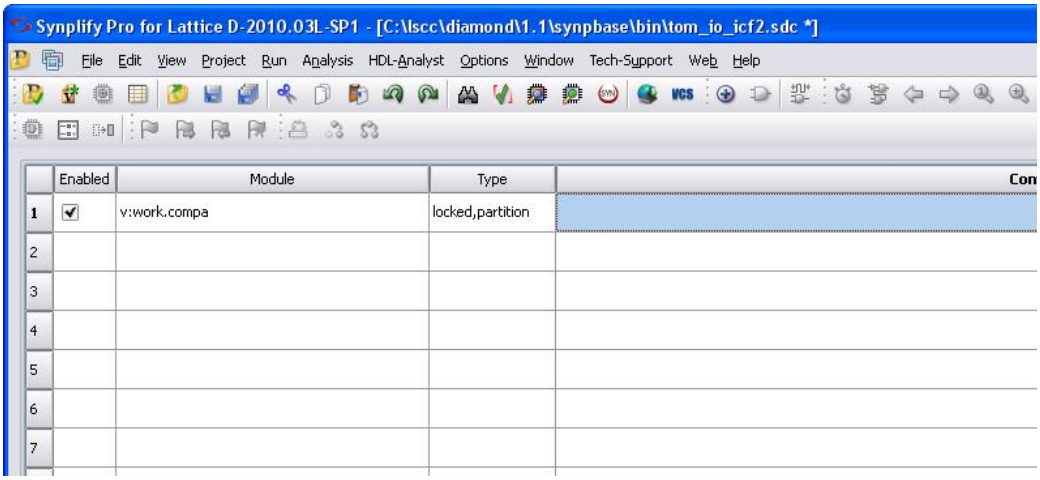
   - Initialize Constraints tab: **Clocks, I/O Delays**
   - Select File Type tab: **Top Level**

   An interactive SCOPE spreadsheet referred to as the Compile Points panel opens. This panel allows you to specify the compile points of your design, and to enable or disable them. Compile points specify design partitions. Note that this panel is only available when defining a top-level constraint file, and only if the device technology supports the compile points.
6. Click on the Maximize button on the top right of the window’s title bar to view the entire SCOPE spreadsheet and choose the **Compile Points** tab.

You now will see a blank spreadsheet with Enabled, Module, Type, and Comment columns.

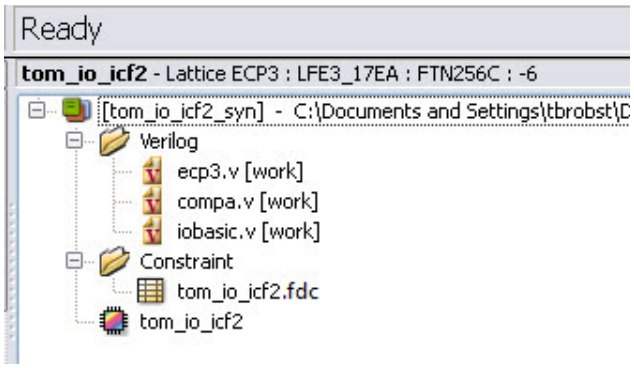
7. Double click on the top row's Module column cell and choose a module that you want it to be a partition from the drop-down menu that appears to the right. In addition, in the Type cell, choose **locked, partition** for each chosen module and ensure that each module is enabled with the check box in the far left column. Repeat these steps for each module that you want it to be a partition in your design.

Figure 2: Creating Locked Partition Constraints in SCOPE Spreadsheet



8. To save your FPGA Design Constraints (.fdc) file in the default synthesis project location, choose **File > Save** and click **Yes** in the prompt dialog.

Figure 3: Project View in Synplify Pro after .fdc File Creation



Please take notice that in the Project view a Constraint folder is created that contains your newly created .fdc file. After you complete this procedure, you should begin the procedure described in the next section "Running Incremental Design in Diamond" on page 658.

Note

You can create an .fdc file using any text editor. It is strongly recommended that you open your .fdc file inside the Synplify Pro constraints editor to check whether the module names are correct. Undefined module names, for example, due to parameterized modules having a name that includes their parameter values, will show in yellow and can be easily corrected here.

See Also ▶"Running the Incremental Design Flow" on page 655

Running Incremental Design in Diamond

After you have created a FPGA Design Constraints (.fdc) file using Synplify Pro as described in "Creating Partitions in Synplify Pro" on page 656, you can begin the procedure of implementing your incremental design in Diamond. This topic describes how to associate that .fdc file with your project, turn on the incremental run modes, and implement your design.

To run the Incremental Design flow in Diamond:

1. Open your incremental design project in Diamond. To enable the incremental design flow for the active implementation, choose **Design > Enable Incremental Design Flow,** or enter the following command in the Tcl Console window at the prompt:

   ```
   prj_incr set -enable
   ```

   The words "Incremental Design Flow" will appear in a yellow box in the upper right corner of the Diamond main window, indicating that the Incremental Design flow is enabled.

   Note

   This procedure presumes that you have completed the first step described in the "Creating Partitions in Synplify Pro" on page 656 topic aand have created an .fdc file to associate with your Diamond project. The Incremental Design flow is only possible if the design includes compile points created during synthesis..

2. Right-click the Synthesis Constraint Files folder in the File List view, and choose **Add**.
3. Choose **Existing File**, and in the Add Existing File dialog box, browse to the .fdc file you generated in the procedure "Creating Partitions in Synplify Pro" on page 656, and click **Add** to target it for use during synthesis. Right-click the added .fdc file, and choose **Set As Active**.
4. In the Process view, double click the **Place & Route Design** process to run the incremental design flow through synthesis, mapping, and place & route.

Be aware that after this point the Map program generates partition database data that contains partition definitions, packing, and optimization strategies that Map uses based on the defined partition.

If this first processing iteration completes successfully, the software automatically creates a backup copy of the placed-and-routed data and stores them in a subfolder called “inc1” inside the Diamond implementation folder. In future design iterations. If PAR completes successfully, these files in “inc1” will be overwritten with the results of the new run and the previous results files in “inc1” are copied to “inc2”, so you have a backup reference design available to you.

Note

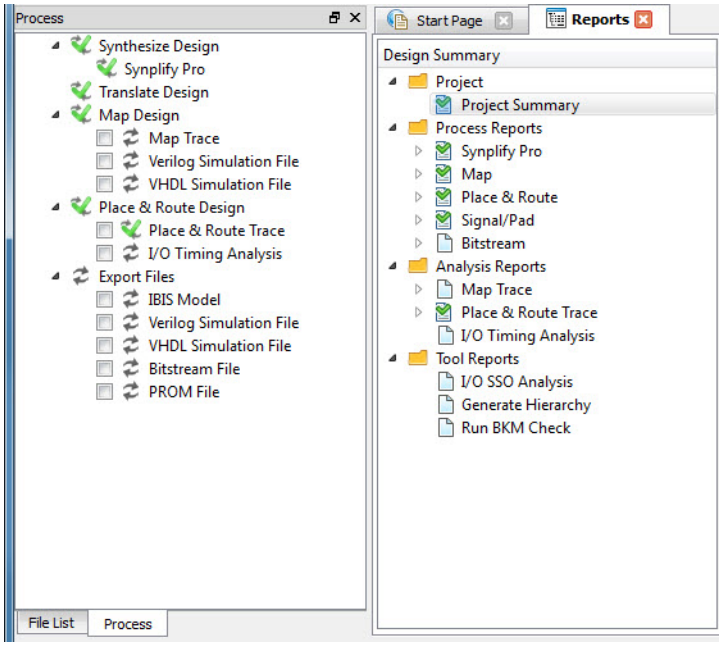
These folders (inc1, inc2, etc.) and files in them are not directly accessible in the Diamond GUI. You must navigate to them using Windows. These folders, and the files in them, are managed by Diamond. You should not manually manipulate them.

For more information on restoring or backing up previous incremental design runs, see the “Backing Up an Incremental Design Run” on page 660 and “Restoring Previous Incremental Design Results” on page 661 topics in this chapter.

It is highly recommended that you use the **File > Archive Project** command (archiving all files) to save the state of your project. See “Archiving Incremental Design Results” on page 662.

5. In the Process view, double click the **Place & Route Trace** process to run post PAR trace and get the timing results on your placed-and-routed NCD file.

Figure 4: Process List Indicating Successful Design Run



6. Use the Report viewer tab in the center panel to open your Map and Place & Route Process Reports; and your Map Trace and Place & Route Trace Analysis Reports.

   To ensure that the results of your design run are satisfactory, see “Viewing Reports for Incremental Design” on page 662, which describes what key elements of the reports you should be viewing.

7. After reviewing the result, you can make necessary HDL code changes, adjust preferences and strategies, set partition controls, and then repeat the process from step 4. above. For setting partition controls in order to fine control the next core processing iteration, use Partition Manager. Refer to “Using Partition Manager” on page 664.

   When you rerun the design flow to start another core processing iteration, the NCD file in “inc1” will be used as the reference design for both Map and PAR. The flow will copy the reference files from “inc1” to the working directory before Map is run.

   Note

   So that Synplify Pro correctly handles partition information, you must stay in your original Diamond project implementation in which you began your incremental design. If you want to create a new implementation and copy source correctly, Synplify will still not reference its partition files. This flow is not ideal for using across multiple project implementations because it relies on a referencing files it uses to guide synthesis and implementation in a relative way.

   Note

   If you perform the **Rerun** process on a synthesized design, or the **Rerun All** process, your synthesis result will be erased and regenerated. Any runtime savings that were gained by preserving the result will be lost, because the preserved result will have been erased.

See Also ▶“Running the Incremental Design Flow” on page 655

▶ “Incremental Design Flow Tcl Commands” on page 2633

Backing Up an Incremental Design Run

You can preserve a “golden” set of PAR reference files from a successful Incremental Design run into a “ref” folder in your Diamond project.

To back up a set of design files in the Incremental Design flow:

- In the Run Manager, right-click on the selected Implementation <strategy > and choose **Incremental > Backup as Golden Reference.**

or

- In the Tcl Command Console, enter the following command:

```
prj_incr backup_golden -impl <implementation name>
```

Performing this task saves a backup of your current files in “inc1” to a newly generated “ref” folder and prevents them from being overwritten on future design runs. This allows you to start your design over again from an ideal starting point that you chose rather than from a possible undesirable point in a later iteration.

See Also ▶See “Running the Incremental Design Flow” on page 655

- “Incremental Design Flow Tcl Commands” on page 2633

Restoring Golden Backup Files

If for some reason you want use the golden backup files as the reference to guide the next core processing iteration, you can restore them.

To restore your golden backup files in the Incremental Design flow:

- In the Run Manager, right-click on the selected Implementation <strategy > and choose **Incremental > Restore Golden as Reference.**

or

- In the Tcl Command Console, enter the following command:

  ```
  prj_incr restore -impl <implementation name> golden
  ```

  Performing this task copies your “golden” backup files in “ref” folder and overwrites your file set in the “inc1” folder, restoring them as the reference for future design runs.

See Also ▶“Running the Incremental Design Flow” on page 655

- “Incremental Design Flow Tcl Commands” on page 2633

Restoring Previous Incremental Design Results

When a core processing iteration completes successfully, this iteration's reference files are copied from /inc1 to /inc2, and this iteration's PAR results are copied to /inc1 and will be used as reference for the next iteration. If for some reason you still want to use the previous reference (in /inc2) to guide the next iteration, you can restore it.

Note

Whenever you run an iteration of your Incremental Design project, your previous results are overwritten in the “inc1” folder in your project implementation directory and moved to the “inc2” folder. However, the software will automatically use whatever is in “inc1” as reference input into its next iteration, not those files in “inc2”.

To restore the previous reference files for Incremental Design:

- In the Run Manager, right-click on the selected Implementation <strategy> and choose **Incremental > Restore Previous Reference.**

or

- Enter the following command in the Diamond Tcl Console window at the prompt:

```
prj_incr restore -impl <implementation name> previous
```

This command restores the previous design run as your reference design in “inc1”, overwriting your last results with the file that was moved to the “inc2” folder. You should have matching file sets in both “inc1” and “inc2” folders.

Note

For the Incremental Design flow to work, the Map process must be rerun on the new reference design. So, restoring a previous result will force the Process List to reset its run status before the Map process.

See Also “Running the Incremental Design Flow” on page 655

- “Incremental Design Flow Tcl Commands” on page 2633

Archiving Incremental Design Results

It is recommended that you use the **File > Archive Project** command to retain the original state of your project. There may be a case in which you might wish to return to your original input source file set or you might want to know what the original file set contained. After many iterations and result files overwrites, it could be difficult to keep track of this.

In addition, you may wish to archive incremental design runs that are close to satisfying your design goals so if future iterations take you further away from your goals you can go back to a point that does not force you to start over.

See Also “Running the Incremental Design Flow” on page 655

Viewing Reports for Incremental Design

This section introduces you to the reports and the specific subsections in them that you should view to evaluate your incremental design run. It also will give you tips and clues to better go about verifying what resources were generated and if they meet expectations.

To view reports for your incremental design:

1. Double click on the appropriate report in either the **Process Report** or **Analysis Report** folder. This assumes that you have run a design flow and that there are generated report files.

2. View the report in the panel to the right and use the scroll bars to navigate to portions of the report that you wish to view.

To use the Timing Analysis view to analyze your incremental design:

1. After you have run through Place & Route design, choose **Tools > Timing Analysis view**.
2. In Timing Analysis view, click on a defined timing preference in the preference window in the lower left corner of the view. The Path Table and the Detailed Path Table spreadsheet information is populated with various data on delay and timing related to that preference.
3. In the bottom right of the Timing Analysis view, click on the **Reports** tab. Notice that as you scroll through the timing report, the elements reported on are highlighted in blue.
4. Right click on some design element in the report and choose either **Show in FP View** or **Show in Physical View** from the popup menu.

See Also ▶"Running the Incremental Design Flow" on page 655

Using Partition Manager

Partition Manager is the Diamond graphical user interface (GUI) used to perform such tasks such as preservation data control, re-implementation effort control, region assignment, and acts as the central interface between the user and partition database.

Partition Manager is only available if Incremental Flow is enabled for an implementation in the project and can be run when the flow is at the pre-Map, post-Map, or post-PAR stage.

Partition Manager can be used after the post-Map stage to create new partitions, edit partition information or remove existing partitions. New partitions can be created for hierarchical modules. You can edit existing partition information such as partition's preservation data level, reimplementation effort, anchor and bounding box. Existing partitions can be deleted and the flow rerun without the deleted partitions.

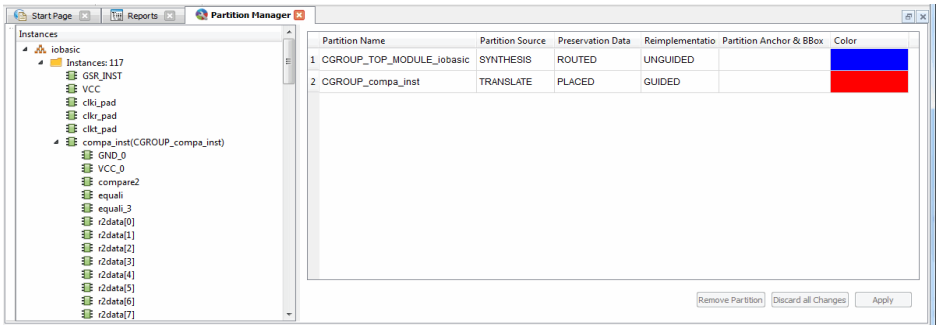
When a partition's information has been updated in either Partition Manager or Floorplan View, the Diamond process is reset to the Translate process.

The GUI for the Partition Manager is in the form of a table. Each row corresponds to a partition in the project. The columns for each row are for the partition name, partition source, the preservation data, the reimplementation effort, the partition anchor and bounding box. All attributes except Partition Name and Partition Source may be edited.

Default values are shown in blue. Anchor and bounding box cells are blank if they are not set for the partition. The top partition module in the project is always in the first row. A color selection box allows you to change the color of the partition in as it appears in Diamond Floorplan View.

The Partition Manager view is shown in Figure 5.

Figure 5: Partition Manager View



An asterisk will appear in the Partition Manager title bar, as shown in Figure 5, if any partition parameters have been modified. Changes can be discarded before they are applied by clicking the **Discard All Changes** button. Changes must be applied by clicking the **Apply** button.

A partition can be added by right-clicking a hierarchical module in the Instances pane on the left side and choosing Create Partition from the pop-up menu.

A partition can be removed by highlighting the partition and clicking the Remove Partition button, or by highlighting a partitioned hierarchical module in the Instances pane on the left side and choosing Remove Partition from the pop-up menu.

Preservation Data Preservation Data allows the user to define partition based implementation process data to be preserved during the next core processing iteration. When it is defined, and if there is no change to invalidate the defined preservation data for the partition, the defined data will be copied from the reference files during the next iteration. All the data after the defined data will be regenerated and the reference is ignored.

Levels of preservation data include:

- Synthesis
- Mapped
- Placed
- Routed

For example, if a partition's preservation data is set to “Mapped”, during the next core processing iteration, if there is no change (such as source code change) to invalidate this partition's mapped result, the new mapped result for this partition will be copied from the reference files. Placed and routed results for this partition will be regenerated regardless of the reference files. If the reference doesn't exist, the new mapped result will be regenerated as well.

Reimplementation Effort If the Preservation Data defined above is invalidated (due to the source code change, for example) and cannot be preserved, Reimplementation Effort allows the user to specify during the next core processing iteration whether the core processes (Map, Placer and Router) should use the reference to guide it (Guided) and generate a new result, or ignore the reference (Unguided) and regenerate a new result.

Preservation Effort has two choices

- Guided
- Unguided

Using the same example as above, if the preservation data was set to “Mapped” and the reimplementation effort was set to “Guided”, during the next core processing iteration:

1. If there is no change to invalidate the defined “mapped” data for the partition, the mapped result will be copied from the reference. Placed and routed results will be regenerated regardless of the reference.
2. If there are changes (such as source code changes) that invalidates the defined “mapped” data for the partition, the reference will be used to “guide” Map to generate a new mapped data. Placed and routed results will be regenerated regardless of the reference.

Similarly, if the preservation data was set to “mapped” and the preservation effort was set to “Unguided”, during the next core processing iteration:

1. If there is no change to invalidate the defined “mapped” data for the partition, the mapped result will be copied from the reference. Placed and routed results will be regenerated regardless of the reference.
2. If there are changes (such as source code changes) that invalidates the defined “mapped” data for the partition, the reference will be ignored. Map will generate new mapped data. Placed and routed results will be regenerated regardless of the reference.

The expected result of the Reimplementation Effort setting, combined with Preservation Data setting, for each partition, is described in Table 1.

Table 1: Preservation Data and Reimplementation Effort Setting Description

Preservation Data Setting	Reimplementation Effort Setting	Expected Result
Synthesis	Guided	Map, place, and route results will be regenerated without using the reference.
Mapped	Guided	If the mapped data is still valid, copy from the reference. Otherwise, use the reference to guide and regenerate mapped data. Placed and routed data will be regenerated regardless of the reference.
Placed	Guided	If the placement data is still valid, copy from the reference, Otherwise, use the reference to guide and regenerate placement data. Routing data will be regenerated regardless of the reference.
Routed	Guided	If the routing data is still valid, copy from the reference. Otherwise, use the reference to guide and regenerate routing data.
Synthesis	Unguided	Map, place, and route results will be regenerated without using the reference
Mapped	Unguided	If the mapped data is still valid, copy from the reference. Otherwise, ignore the reference and regenerate map data. Placed and routed data will be regenerated regardless of the reference.
Placed	Unguided	If the placement data is still valid, copy from the reference. Otherwise, ignore the reference and regenerate placement data. Routing data will be regenerated regardless of the reference.
Routed	Unguided	If the routing data is still valid, copy from the reference. Otherwise, ignore the reference and regenerate routing data.

Preservation Data and Reimplementation Effort settings control only Map, Place and Route. The synthesis process is not controlled by these settings. Synthesis is controlled by Synplify Pro.

Changing only preservation data and/or reimplementation effort settings will not reset the flow. To force any changes to take effect without changing the source code, in the Diamond Process view, right-click on Place and Route Design and choose **Rerun All**.

The default preservation data setting is “Placed.” The default reimplementation effort setting is “Guided.”

Anchor and Bounding Box The partition's anchor and/or bounding box (BBox) are editable with the Partition Anchor and BBox dialog box. The dialog box specifies, based on the device, acceptable values for row, column, height, and width.

After making the desired changes to the partitions, the changes will be applied to the implementation. If the anchor or bounding box changes were made after a successful PAR, the process will be reset to before PAR.

When the user closes the Partition Manager, if changes were made but not applied or reverted, a dialog appears asking “Do you want to save your in-memory ICF changes into <*file_name*>.icf?” with the options of **Yes**, **No**, and **Cancel**.

When the user closes Diamond, if changes were made but not applied or reverted, a dialog appears listing changes that need to be saved. The user may select the file if he wants changes to be saved to it before closing.

Running the Partition Manager

If you have an active implementation with incremental design flow enabled, after synthesis and translation, you can run the Partition Manager.

To run the Partition Manager:

1. Diamond, choose **Tools > Partition Manager** or click the  toolbar button.

See Also

- “Creating a New Partition with the Partition Manager” on page 667
- “Setting Preservation Data” on page 668
- “Setting Reimplementation Effort” on page 669
- “Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time” on page 669
- “Specifying Partition Anchor and Bounding Box” on page 670
- “Changing Partition Colors” on page 671
- “Cross-Probing Partitions from Partition Manager” on page 672
- “Removing a Partition Using Partition Manager” on page 672

Creating a New Partition with the Partition Manager

You can create a new partition to hierarchical modules in an incremental design after the MAP process has been run.

To create a new partition with Partition Manager:

1. Right-click a hierarchical module in the Instances pane on the left side and choose **Create Partition** from the pop-up menu.
2. In the Partition Anchor and BBox dialog box, you can either accept the default Anchor and Bounding box settings, or you can specify Anchor and Bounding box locations, and then click **OK**.

See Also

- “Setting Preservation Data” on page 668
- “Setting Reimplementation Effort” on page 669
- “Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time” on page 669
- “Specifying Partition Anchor and Bounding Box” on page 670
- “Changing Partition Colors” on page 671
- “Cross-Probing Partitions from Partition Manager” on page 672
- “Removing a Partition Using Partition Manager” on page 672

Setting Preservation Data

You can set the partition data to be preserved and used for the next incremental iteration in Partition Manager. Preservation data includes partition processing results including synthesized data, mapped data, placement data and routed data.

To set Preservation Data in Partition Manager:

1. Double-click the Preservation Data cell of the Partition Name row you wish to set. You can also right-click on the cell to display a menu with the preservation data options. This menu has a “Clear” option to reset the preservation data level to the default value.
2. In the drop-down menu, select from **Synthesis**, **Mapped**, **Placed**, or **Routed** data you wish to preserve.
3. Click **Apply** when finished editing. To discard all changes and revert to the last save, click **Discard All Changes**.

See Also

- “Running the Partition Manager” on page 667
- “Creating a New Partition with the Partition Manager” on page 667
- “Setting Reimplementation Effort” on page 669
- “Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time” on page 669
- “Specifying Partition Anchor and Bounding Box” on page 670
- “Changing Partition Colors” on page 671

- “Cross-Probing Partitions from Partition Manager” on page 672
- “Removing a Partition Using Partition Manager” on page 672

Setting Reimplementation Effort

You can set the partition effort for changed partitions to determine how previous processing results are used during the current incremental iteration.

To set Reimplementation Effort in Partition Manager:

1. Double-click the Reimplementation Effort cell of the Partition Name row you wish to set. You can also right-click on the cell to display a menu with the reimplementation effort options. This menu has a “Clear” option to reset the reimplementation effort level to the default value.
2. In the drop-down menu, select from **Guided** or **Unguided**.
3. Click **Apply** when finished editing. To discard all changes and revert to the last save, click **Discard All Changes**.

See Also

- “Running the Partition Manager” on page 667
- “Creating a New Partition with the Partition Manager” on page 667
- “Setting Preservation Data” on page 668
- “Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time” on page 669
- “Specifying Partition Anchor and Bounding Box” on page 670
- “Changing Partition Colors” on page 671
- “Cross-Probing Partitions from Partition Manager” on page 672
- “Removing a Partition Using Partition Manager” on page 672

Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time

You can set the partition effort or Reimplementation Effort for some or all partitions at the same time.

To set some or all partitions to same Preservation Data or Reimplementation Effort in Partition Manager:

1. Select some items, or all items, from the Preservation Data column or Reimplementation column, and right-click.
2. In the pop-up menu, choose one of the following:

- For Preservation Data, select from **Synthesis**, **Mapped**, **Placed**, or **Routed** data you wish to preserve.
- For Reimplementation Effort, select from **Guided** or **Unguided**.

3. Click **Apply** when finished editing. To discard all changes and revert to the last save, click **Discard All Changes**.

See Also

- "Running the Partition Manager" on page 667
- "Creating a New Partition with the Partition Manager" on page 667
- "Setting Preservation Data" on page 668
- "Setting Reimplementation Effort" on page 669
- "Specifying Partition Anchor and Bounding Box" on page 670
- "Changing Partition Colors" on page 671
- "Cross-Probing Partitions from Partition Manager" on page 672
- "Removing a Partition Using Partition Manager" on page 672

Specifying Partition Anchor and Bounding Box

You can set the partition anchor and bounding box within the device architecture using Partition Manager. As values are changed, the number of LUTs, REGs, and EBRs encompassed by the bounding box will be shown in the dialog.

To set specify partition anchor and/or bounding box:

1. Double-click the Partition Anchor & BBox cell of the Partition Name row you wish to set, or right click the cell and choose Edit Anchor and BBox, to display the Position Anchor and BBox dialog box.
2. If modifying anchor, check the **Anchor** box and specify Row and/or Column.
3. If modifying bounding box, check the **BBox** box and specify Height and/or Width.
4. Click **OK**

When a partition has been edited and applied in Partition Manager, the Diamond process is re-set to the pre-Map process**.**

Note

Anchor and bounding box can also be edited in the Floorplan View. Refer to "Editing Partitions in Floorplan View" on page 675.

See Also

- “Running the Partition Manager” on page 667
- “Creating a New Partition with the Partition Manager” on page 667
- “Setting Preservation Data” on page 668
- “Setting Reimplementation Effort” on page 669
- “Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time” on page 669
- “Changing Partition Colors” on page 671
- “Cross-Probing Partitions from Partition Manager” on page 672
- “Editing Partitions in Floorplan View” on page 675
- “Removing a Partition Using Partition Manager” on page 672

Changing Partition Colors

You can change the partition colors that display in Diamond Floorplan View by using the Color selection box in Partition Manager.

To change the color of a partition in Partition Manager:

1. Double-click the item you want to change. The Select Color dialog box opens.
2. Select a color by doing one of the following:
   - Click one of the basic colors on the left.
   - Click in the color pallet on the right.
   - Enter values in the boxes at the bottom right. Work in one column or the other as they are two separate methods of coding color.

     The rectangle in the lower-middle changes to show the selected color. You can change the darkness with the vertical slider at the upper-left.
3. Click **OK**.

See Also

- “Running the Partition Manager” on page 667
- “Creating a New Partition with the Partition Manager” on page 667
- “Setting Preservation Data” on page 668
- “Setting Reimplementation Effort” on page 669
- “Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time” on page 669
- “Specifying Partition Anchor and Bounding Box” on page 670
- “Cross-Probing Partitions from Partition Manager” on page 672
- “Removing a Partition Using Partition Manager” on page 672

Cross-Probing Partitions from Partition Manager

You can cross probe and view partitions in either Diamond Floorplan View or Physical View from Partition Manager. In order to perform cross probing, the partitions must have anchor and bounding box set.

To cross probe a partition in Floorplan View from Partition Manager:

1. In Partition Manager, highlight the Function Name, Preservation Data, Reimplementation, Partition Anchor & BBox, or Color box in a partition.
2. Right-click, and in the dropdown menu, choose either **Show In > Floorplan View** or **Show In > Physical View**.

See Also

- "Displaying Partitions in Floorplan View" on page 674
- "Displaying Partitions in Physical View" on page 675
- "Running the Partition Manager" on page 667
- "Creating a New Partition with the Partition Manager" on page 667
- "Setting Preservation Data" on page 668
- "Setting Reimplementation Effort" on page 669
- "Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time" on page 669
- "Specifying Partition Anchor and Bounding Box" on page 670
- "Changing Partition Colors" on page 671
- "Cross-Probing Partitions from Partition Manager" on page 672
- "Removing a Partition Using Partition Manager" on page 672

Removing a Partition Using Partition Manager

You can remove a partition using Partition Manager.

To remove using Partition Manager:

1. In Partition Manager, highlight the partition you wish to remove.
2. Click **Remove Partition,** then click **Apply**.

   or:

   Right-click a partitioned hierarchical module in the Instances pane on the left side and choose **Remove Partition** from the pop-up menu.

   The partition will be removed when the Translate process is re-run.

See Also

- “Running the Partition Manager” on page 667
- “Creating a New Partition with the Partition Manager” on page 667
- “Setting Preservation Data” on page 668
- “Setting Reimplementation Effort” on page 669
- “Setting Preservation Data or Reimplementation Effort for Some or All Partitions at the Same Time” on page 669
- “Specifying Partition Anchor and Bounding Box” on page 670
- “Changing Partition Colors” on page 671
- “Cross-Probing Partitions from Partition Manager” on page 672

Floorplan View and Physical View Partition Support

Both Floorplan View and Physical View can display partition information to support the Incremental Design Flow. Partitions can be edited in Floorplan View, but not in Partition View.

- Floorplan View loads partition information that has not yet been loaded into memory. Data is only loaded after the Translate process. See “Displaying Partitions in Floorplan View” on page 674.
- Physical View can display partition information to support the Diamond Incremental Design Flow. Data is only loaded after the Translate process. See “Displaying Partitions in Physical View” on page 675.

Displaying Partitions in Floorplan View

Partitions are displayed nearly identically to UGROUPs and REGIONs in Floorplan View. If a partition has an anchor and bounding box, then it can be displayed in Floorplan View. However, if a partition does not have either an anchor or bounding box, the partition will not be displayed in Floorplan View. Refer to “Specifying Partition Anchor and Bounding Box” on page 670 for information on how to specify anchor and bounding box.

Partitions will have a solid border with a patterned background, similar to UGROUP bounding boxes. The patterned background is different than that of a UGROUP.

Note

REGIONs have a solid background.

To display partitions in Floorplan View:

1. Choose **Tools > Floorplan View.**
2. By default, partitions are displayed in Floorplan View. If they do not appear on the layout, click the  button on the vertical toolbar, or click **View > Preference Placement Display > Display Partitions**.

See Also

- “Editing Partitions in Floorplan View” on page 675
- “Cross-Probing Partitions from Partition Manager” on page 672
- “Displaying Partitions in Physical View” on page 675

Editing Partitions in Floorplan View

If an anchor and bounding box have been specified for a partition in Partition Manager, the anchors and bounding boxes can be viewed and edited in Floorplan View.

To edit partitions in Floorplan View:

1. Choose **Tools > Floorplan View.**
2. By default, partitions are displayed in Floorplan View. If they do not appear on the layout, click the button on the vertical toolbar, or click **View > Preference Placement Display > Display Partitions**.
3. Right-click the partition you wish to edit and chose **Edit Partition** to display the Position Anchor and BBox dialog box.
4. If modifying anchor, check the **Anchor** box and specify Row and/or Column.
5. If modifying bounding box, check the **BBox** box and specify Height and/or Width.
6. Click the Save Partitions button on the vertical toolbar.

When a partition has been edited and applied in Floorplan View, the Diamond process is re-set to the pre-Map process**.**

Note

Anchor and bounding box can also be edited in the Partition Manager. Refer to "Specifying Partition Anchor and Bounding Box" on page 670.

See Also

- "Displaying Partitions in Floorplan View" on page 674
- "Cross-Probing Partitions from Partition Manager" on page 672
- "Displaying Partitions in Physical View" on page 675

Displaying Partitions in Physical View

Physical View displays partition information with the similar way as UGOUP or REGION. The color used for partitions are different than that of the UGROUP or REGION.

To display partitions in Physical View:

1. Choose **Tools > Physical View.**
2. By default, partitions are displayed in Physical View. If they do not appear on the layout, click **View > Show Layers > Partition**.

See Also

- “Displaying Partitions in Floorplan View” on page 674
- “Editing Partitions in Floorplan View” on page 675
- “Cross-Probing Partitions from Partition Manager” on page 672

Recommended Strategies for Incremental Design

This chapter provides recommended strategies for using Incremental Design. It outlines the circumstances that can dictate when using this flow is the best approach and when it is not. For the most part, what might lead you to use this flow will likely be a matter of trial and error, but it is useful to gain an understanding as to when Incremental Design should be your first approach to implementing your design.

Although incremental flow in general will provide shorter runtime and relatively consistent result, there will be situations it may take longer. For example, it might be possible that incremental flow can not meet the timing goals while normal flow can. Incremental flow is suitable for well-partitioned designs. Flat designs or designs with many critical paths crossing partitions may actually result in a drop-off in performance using the Incremental Design flow.

The following are common terms used in incremental design flow:

Unguided iteration A core processing (Map, Place and Route) iteration where the physical reference design (NCD) doesn't exist, or it does exist but is ignored. To run an unguided iteration if the reference NCD is available but needs to be ignored, set all partitions' preservation data to “Synthesis” in Partition Manager

Guided iteration A core processing (Map, place and route) iteration where the reference NCD is used for all partitions or certain partitions, based on the user settings through the Partition Manager.

ICF file The partition data base/file that holds all of the partition’s related information and user settings set through the Partition Manager. Each reference NCD has its own ICF file associated with it.

Recommended Incremental Design Flow: Scenario 1

This scenario applies to designs in which timing closure can be easily achieved, or has been previously addressed using incremental flow by locking down the timing critical partitions. The majority of the of design is locked down, meaning no further source code changes are allowed, and the number of partitions that need to be changed are very limited.

1. In Diamond, create a new project and implementation, and add HDL codes.
2. Add synthesis constraints and adjust synthesis strategies if necessary, and complete a synthesis iteration.
3. Open the Hierarchy window, examine resource usage information for each module in order to guide partition creation.

4. Launch Synplify Pro, compile the design, then define compile points based on the observations in step 3.
5. Adjust synthesis, Map and PAR strategies if necessary, and complete synthesis, Map and PAR.
6. Examine the process reports. For these types of designs, timing closure should be easily achieved.

   Optionally, use Partition Manager to change Preservation Data of all partitions to **Routed** or **Placed** (default) and leave Reimplimentation Effort as **Guided** (default).
7. If required, modify HDL code to certain partitions, and repeat step 5.
8. If required, repeat step 6 and step 7.

For these types of designs, timing closure can be easily achieved and runtime reductions for each iteration should be obvious.

Recommended Incremental Design Flow: Scenario 2

This scenario applies to designs in which timing closure can be easily achieved for certain part of a design but not in other parts of the design. For example, timing critical paths fall into certain partitions.

1. In Diamond, create a new project and implementation, and add HDL codes.
2. Add synthesis constraints and adjust synthesis strategies if necessary, and complete a synthesis iteration.
3. Open the Hierarchy window and examine resource usage information for each module in order to guide partition creation.
4. Launch Synplify Pro, compile the design, then define compile points based on the observations in step 3.
5. Adjust synthesis, Map and PAR strategies. If necessary; it is recommended that:
   a. Multi-seeds PAR is enabled.
   b. Routing method option is set to NBR.
6. Complete synthesis, Map and PAR.
7. Examine the process reports. Find timing critical paths and identify partitions that have timing problems. For these types of designs, timing problems should fall into certain partitions but not the others.
8. Using Partition Manager:
   a. For partitions with identified timing problems, set Preservation Data to **Synthesis**.

   b. For partitions without timing issue, set Preservation Data to **Routed** or **Placed** (default) and leave Reimplementation Effort to **Guided** (default).

9. If required, modify HDL code to identified partitions in order to improve the timing
10. Repeat step 5 and step 6
11. If required, repeat step 7 through step 10
12. When all timing issues are fixed, the design will be in a condition as described in “Recommended Incremental Design Flow: Scenario 1” on page 677. At this point, Partition Manager can be used to set all partitions Preservation Effort to **Routed** or **Placed** (default), and Reimplementation Effort to **Guided**.

    If there is any need to do a minor change to certain partitions, we can follow the step 7 of “Recommended Incremental Design Flow: Scenario 1” on page 677.

Recommended Incremental Design Flow: Scenario 3

This scenario applies to designs in which timing closure is hard to achieve for the whole design and partitions need to be redefined in order to isolate critical paths

1. In Diamond, create a new project and implementation, and add HDL code.
2. Add synthesis constraints, adjust. synthesis strategies if necessary, and complete a synthesis iteration
3. Open the Hierarchy window and examine resource usage information for each module in order to guide partition creation.
4. Launch Synplify Pro, compile the design, and define compile points based on the observations in step 3.
5. Adjust synthesis, Map and PAR strategies if necessary. It is recommended that:

   a. Multi-seeds PAR is enabled.

   b. Routing method option is set to NBR.

6. Complete synthesis, Map and PAR.
7. Examine the process reports and identify timing critical paths. For these types of designs, there may be more than one critical path across multiple partitions. Isolate each critical path, identify modules that are included in the path, and use the information as the guideline for partition modification.
8. Using Partition Manager, set all partition Preservation Data to **Synthesis**.
9. If required, modify HDL code to identified critical paths in order to improve timing.

10. Based on the observation in step 7, you might need to change the partition definitions. Using Synplify Pro, add or remove compile points, as needed. It may be necessary to group certain modules that cover a critical path to create a new module to be able to define a new compile point to cover a complete critical path. In some cases, you might also want to set the partitions without timing issue with Preservation Data to **Routed** or **Placed** (default) and leave Reimplementation Effort to **Guided** (default)
11. Repeat step 5 and step 6.
12. If required, repeat step 7 through step 11
13. When all timing issues are fixed, the design will be in a condition as described in “Recommended Incremental Design Flow: Scenario 1” on page 677, we can use Partition Manager to set all partitions Preservation Data to **Routed** or **Placed** (default), and the Reimplementation Effort to **Guided**.

    If there is any need to do a minor change to certain partitions, we can follow the step 7 of “Recommended Incremental Design Flow: Scenario 1” on page 677.

General Incremental Design Flow Limitations

The following is a list of general limitations that you should use as a guideline for determining whether or not your design is a good candidate for running through the Incremental Design flow.

- Although the Incremental Design flow in general will provide shorter runtimes and a relatively consistent result, there will be situations it may take longer. For example, in some cases the incremental flow might not meet timing goals whereas the normal flow can.
- The Incremental Design flow is suitable for well-partitioned designs. A flat design or a design with many critical paths crossing partitions may actually result in poorer performance using the incremental flow.
- Designs that have a very high resource utilization may not be a good candidates for running in the Incremental Design flow because typically this flow requires ample resource availability.
- Designs with congestion may also not be good candidates for the Incremental Design flow for the same reason, that is, incremental designs tend to require more resource availability to resolve congestion, such as loose placement and detour routing.
- The partition floorplanning that occurs in this flow can automatically generate partition constraints. A completed design is required to generate a reasonable partition. Running incremental flow on partial design is not recommended. Oftentimes the auto-generated partition constraint is not optimal. It is strongly recommended that if you must adjust partition constraints, that you do so based on your knowledge of the design.
- To achieve the best stability and quick timing loop closure, it is strongly recommended that you keep the changes from one iteration to the next

iteration to a minimum. Any major change or a lot of minor changes will result in a completely different result which may undermine the overall effectiveness of the incremental design flow's inherent strategy.

- The Incremental Design flow may yield loose packing and lesser performance due to partition constraints. Also, if the incremental router is applied, longer runtime and worse performance may result, if, from iteration to iteration, the change is significant or some congestion occurs.
- Reveal is not generally recommended for use with the Incremental Design Flow. Any module that Reveal needs to trigger or trace into will be a changed module, so its implementation will not be preserved from the reference design. Additionally, it is likely that these debug connections will cross multiple partitions.
- The Timing Driven Mapping (-tdm), Timing Driven Packing (-td_pack), and Timing Driven Node Replication (-split_node) Map strategy options are not supported in the Incremental Design flow. These timing-driven options are not supported because they do not honor design partition region boundaries.

General Synthesis Recommendations

This section lists any general recommendations to ensure the Incremental Design flow can be run in the Diamond environment and to eliminate or minimize any error or warning messages you encounter.

The synthesis recommendations for Incremental Design are as follows:

- A minor RTL change may result in significant difference in your mapping and place-and-routing result. For this reason, it is recommended that you make any HDL source changes before using the incremental design flow, or certainly not when you are using the guided iteration run flow mode that uses an NCD as a reference file to guide implementation. To set the next iteration as unguided iteration, change all of the partitions preservation data to "Synthesis" using Partition Manager.
- To avoid issues during synthesis, you must use Synplify Pro version of the Synplify software since Compile Points feature is not supported in other versions. The Synplify Pro version in Diamond supports the Incremental Design flow.
- In the beginning of the Incremental Design flow you must create a top-level FPGA Design Constraints (.fdc) file in Synplify Pro and associate it with the Diamond project you are running. If you do not do this, the Incremental Design will not work properly and the flow is actually running as in a normal flow.
- Create your .fdc file in the graphical user interface of the Synplify Pro tool, not the command line. If you do not do this, module names to apply your compile points types (locked,partition) may not be recognizable to you in the command line because the modules are parameterized and appear with names such as "ro_cnt_8s_0_6s" and the like. All of these extra prefixes or suffixes like this are the parameterization.

- To discover and analyze error and warning conditions for your partitioned modules, open Synplify Pro (synplify_pro) and open the SCOPE editor panel which allows you to edit your .fdc file. Click on the Compile Points tab and observe that all of the compile points are marked in yellow, which indicates an error or warning condition.

Avoiding Common Design Flow Issues

You should go about using the Incremental Design flow as prescribed in this guide in a step-by-step fashion to ensure that you have initiated the flow correctly and you get the expected outputs when you run it. It is probable in these cases you have not performed a step and this has prevented Diamond from recognizing your active implementation of your project as an incremental design.

See "Troubleshooting Error and Warning Messages" on page 685 on diagnosing common usage issues.

Defining Design Partitions How you define your partitions is design-dependent and will have an impact on how effective using the Incremental Design flow in Diamond will be. This section provides you with useful information you can employ when you are defining partitions with compile points when you are creating your .fdc file in Synplify Pro.

Below is a list of factors involved in defining partitions that can effect the quality of results you might achieve using the Incremental Design flow. The list also includes how partitions are treated in regard to resource utilization.

The following describe how partitions can affect the incremental design flow:

Number of Partitions It is recommended that you limit the number of partitions in your design to a reasonable figure, for example, 30, with respect to the percent partitions in total utilization which is described later in this section. If your design has too many partitions it will lose the global placement picture. Also, it is best to contain the portions of the design that could change to the smallest partition possible, thus preserving other parts of the design. As a general rule, if more than about 10 percent of the design is changing, it is usually advisable to re-run an unguided incremental pass.

As a recommended practice, you should not change the number of partitions in your design when running guided iterations, or you may encounter unexpected results. Defining what gets partitioned in your design should be done prior to using an NCD reference design for more incremental changes.

Size of Partitions There is no size limit for a design partition. In general, partition size is defined by a given functional module where a compile point is applied. However, any changes to a large partition can result in significant changes in how Map and PAR pack, place and route components and hence your Fmax result as well.

Shape of Partition Regions A partition's region is always in the shape of a rectangle. A non-rectangular shape is not allowed. However, overlapping among partitions is allowed in the Incremental Design flow to achieve optimal results.

Percent Partitions in Total Utilization It is suggested that at least 75 percent of your incremental design is covered by partition instances. To find out what percentage of your design is covered by partitions, look at the "PARTITION Utilization" section in the Map Report (.mrp) file. This section of the file is shown below to illustrate how this percentage is given along with the total number of design partitions:

```
Total comps 450 (92.78%) in 65 partitions.
```

Please understand that the limit on the number of design partitions mentioned previously in this section is a more critical factor in the successful implementation of your incremental design.

Components within Partitions Understand that partition BBOX definitions keep those components in a particular design partition confined to a region that is SLICE-based but the region will not include in its boundaries the placement of special elements such as EBRs, DSP, and DDR because they are row-based. However, EBR and DSP will be placed close to a relative partition.

So, the BBOX definition does not affect the placement of these special elements at all, only logic that is placed in SLICE components or PFU/PFF elements. If you want to place special elements within a specific physical region, use the UGROUP or LOCATE preferences as appropriate. In addition, due to architecture limitations, not all global resources will be able to be included in an individual partition. For example, global resources such as PLL, DLL, PCS, DCS, CLKDIV, and DLLDEL may not be used in the same partitions due to complicated rules and shape of ring. PIO/IOL (particularly clock PIO) are considered a global resource as well, which means PIO/IOL will not be included in a partition.

Note

Any conflict between a user preference and a partition definition must be resolved by the user. The placer will simply error out when encounters a conflict, such as a PGROUP or carry-chain crossing different partitions.

Placement Variability in Guided Iterations It is important to understand how guided iterations use the reference design to guide placement. The major concept to understand is that when you are using the incremental flow, the default settings in the data file preserve placement of the reference design unless the design's source HDL changes. This is implemented that way by design.

So, you need to be aware that a poor placement result will be preserved as readily as a good placement result. For this reason, we recommend that you get your design close to timing closure in the unguided iteration run flow mode before switching to the guided iteration run flow mode.

You should expect to see no variation in timing inside a partition with unchanged HDL in incremental flow when you run multiple cost tables with default data file settings. Also, if you attempt to change placement in the guided iteration by changing UGROUP or LOCATE preferences, you may see unexpected results. See “Troubleshooting Error and Warning Messages” on page 685 for related issues.

Handling Timing Issues in Guided Iterations If you are already in guided iteration and are seeing timing errors reported in your TRACE report, you have the following three options:

- Go back to unguided iteration run flow mode. This would be the appropriate choice if you believe changes to the partition planning or major HDL changes throughout the design are required.
- If the timing errors can be addressed by making HDL changes (e.g., adding pipeline stages), stay in guided run flow mode and use default preservation data and re-implementation effort settings for all partitions.
- If you believe the timing errors could be addressed by allowing some variability into the placement of a partition where HDL is not changed, you can achieve this by changing settings through the Partition Manager.

  To do this, open Partition Manager, and for all partitions that you want variability, change their preservation data to “mapped” and reimplementation effort to ‘guided.” Then, run multiple cost tables and force a PAR rerun. After doing this you will see some variability in the result for that partition within the constraints of the partition planning that appear in the data file.

Adding Pipeline Registers to Incremental Designs The Incremental Design flow is well suited to the case where you want to add pipeline registers and keep the rest of the design unchanged for placement. In guided iteration run flow mode, add the registers to your HDL and rerun the flow. No manual intervention should be necessary.

If for some reason you require that the new registers go into a specific area that is within their partition boundary and that has enough open sites to place the new registers, you can do this. In this particular case, add a UGROUP to the new registers with bounding box (BBOX) and anchor point (SITE). Note that this is only an option with changed logic. For more information, see the troubleshooting section on UGROUPs in incremental mode.

Troubleshooting Error and Warning Messages

This chapter provides a listing of error and warning messages that are specific to running the Incremental Design flow in Diamond. The chapter also includes a list of current known issues that you might encounter during the flow and provides some information to help you work around them effectively.

Troubleshooting Missing Files and Folders This section provides some examples of common issues that occur to users who are missing some file or folder inherent to the Incremental Design flow and they cannot understand the cause of it and cannot proceed.

An Incremental Design flow user could encounter cases where they expect that there is some file or folder in a certain location, but for some reason it does not exist. In most cases, this happens because they have skipped an important step in the design flow steps as they are described in “Running the Incremental Design Flow” on page 655

The following is a list of the most common reasons why you are not seeing the files and folders you would expect to see in the Incremental Design flow:

- You did not create an FPGA Design Constraints (.fdc) file in Synplify Pro.
- You did not create any compile point constraints in your .fdc file in Synplify Pro.
- You did not associate your .fdc file with your Diamond project.
- You did not enable the Incremental Design flow.

Known Issues The section lists known issues that you might encounter in the Incremental Design flow. If there is a workaround, that information is provided.

Extra partition is defined by Incremental Design flow In some cases, after Synthesis, an extra partition is defined by the Incremental Design flow. For example, if you have 10 compile points defined in Synplify Pro, the Map or PAR may report more than 10 partitions. There are two possible reasons:

- The top level design is always treated as a partition.
- Synplify Pro compile points are module based. Therefore, if a module contains a compile point defined with “n” number of instances, then effectively there will be “n” number of partitions for that module.

Missing SITE/BBOX definition for partition in ICF Sometimes after running, the PAF utility does not assign an anchor SITE and regional BBOX to the PARTITION setting in the ICF file. In these cases you will see a message in the .par file (i.e., if the PAF is run inside par as in Diamond) similar to the following:

```
INFO: PARTITION CGROUP_serdes_quad has no valid comp, this
partition will be ignored.
```

And then later,

```
PARTITION 2 : “CGROUP_serdes_quad” (SITE and BBOX are not
assigned)
```

There will be no usage information shown in this case because there are no slices in the partition.

Issue: One reason PAF does not define a SITE and BBOX for a given partition in the ICF is that your compile point constraint does not include any SLICEs. This could happen for the following reasons:

- They are optimized away. This may be expected behavior, but also may happen if inputs or outputs do not connect to ports. check tool logs)
- There are actually no contents of the partition that are slices. This could happen, for example, if you put a compile point on a module which only contains a SERDES block, or only I/O. This is not an error condition but a compile point like this does serve any purpose for Map and PAR.

Solution: Check your tool log files in Synplify and attempt to determine why your inputs and outputs are not connecting to ports.

Using I/O buffers with tristate controls in modules with compile points If I/O buffers with tristate controls are present within modules with compile points defined in your design, you will encounter error messages during implementation in Diamond.

Issue: The synplify_pro synthesis tool cannot support I/O buffers with tristate controls inside of modules with compile points. If the buffers are instantiated, it will detect them, issue a warning message, and disable the compile point. Check the Synplify log or the warning tab in Diamond. The message will appear similar to the following example shown below:

```
@W: BN106 |Cannot apply constraint syn_compile_point to
v:work.immir

@W: BN106 |Cannot apply constraint syn_cptype to v:work.immir
```

If you miss this warning message, you will only see this after the Map process, when you are missing partitions in the Map Report (.mrp) PARTITION report and the ICF file.

Solution: Either do not use a compile point on the module or move the buffers up to the top level of the design's hierarchy in the HDL.

Unused design elements may generate Map errors in modules with compile points Using compile points with designs that contain unused logic may result in the unintended functionality and Map error messages.

Issue: If you use compile points during synthesis and your design contains unused elements, for example, a multiplier exists in a test case that is driven by a driver in another partition, but the outputs are not intended to be used, it will likely result in Map errors with messages similar to the following.

```
ERROR - map: Mico32_u/LM32/cpu/multiplier/product_2_36_71 : Pin
A13 has no driver. Possible causes are (1) redundant logic or
(2) undriven input.

ERROR - map: Mico32_u/LM32/cpu/multiplier/product_2_36_71 :
Input B Pin B0 is not connected but the Source is statically
connected to GND(Parallel Input)
```

Solution: Either do not use a compile point on the module or remove any unused logic from your design before mapping.

Changing I/O placement and function The Incremental Design flow tries to preserve the I/O placement of the reference design. In the incremental flow, you can change I/O placement via new LOCATE preferences, where the new placement is feasible – that is, where the pins are available, and support the function of the I/O.

The new preferences will be honored and you will see messages, as shown below, in the PAR Report (.par) file that inform you that the placement in the reference design is being overridden by the new LOCATE preferences.

```
WARNING - par: Comp 're' has been LOCATE'd and will not be
placed as per guided file
```

Issue: Changing I/O function in the incremental flow can potentially cause an I/O placement failure. An example of a problem like this might be changing a non-DDR I/O to a DDR, or changing an I/O type from LVCMOS to LVDS. The original placement may no longer be feasible due to the resource availability at the reference design's pin locations.

Solution: If you see an I/O placement failure in the Incremental Design flow, change your LPF preference file to manually locate the changed I/O to the appropriate pins or, instead of using the incremental flow, use the normal run flow mode.

Parameterized modules in standalone Synplify Pro Using the standalone Synplify Pro software for Incremental Design, parameterized modules in a design make it impossible to determine what their actual module names are and results in error messages later in the design flow that are difficult to interpret.

Issue: In this specific case, a user encountered an Map error that prevented the ICF file from being opened which occurred using an unguided iteration run flow mode they attempted to process through Diamond's Place & Route Design process. The following Map error message was reported in the Output view:

```
WARNING - map: There is no partition, and the option "-inc" is
ignored.
```

This warning implies that there were no PARTITION properties or constraints recognized by Map so they were not passed through the implementation flow. The .fdc file contains compile points that become PARTITION settings in the Incremental Design flow.

This condition occurred when the user attempted build their .fdc constraints file in a standalone Synplify Pro software and because of parameterized module naming not matching the actual module names, it appears to the software that there were no compile points. In the Synplify Report the following "Cannot apply constraint" error message occurred:

In addition, if you reopen the standalone version of Synplify Pro, choose **Run > Compile Only** and open the .fdc file using the SCOPE editor spreadsheet

```
@W: BN106 |Cannot apply constraint syn_compile_point to v:work.ud_cnt
@W: BN106 |Cannot apply constraint syn_cptype to v:work.ud_cnt
@W: BN106 |Cannot apply constraint syn_compile_point to v:work.ro_cnt
@W: BN106 |Cannot apply constraint syn_cptype to v:work.ro_cnt
@W: BN106 |Cannot apply constraint syn_compile_point to v:work.ud_cnt_uniq_1
@W: BN106 |Cannot apply constraint syn_cptype to v:work.ud_cnt_uniq_1
@W: BN106 |Cannot apply constraint syn_compile_point to v:work.ro_cnt_uniq_2
@W: BN106 |Cannot apply constraint syn_cptype to v:work.ro_cnt_uniq_2
```

panel. Now, click on the Compile Points tab and observe that all of the compile points are marked in yellow. This indicates that there is an error condition you have to resolve before you proceed.

Solution: You should create your .fdc file inside of the Synplify Pro graphical user interface and it is highly recommended that you run Synplify Pro for Lattice from within the Diamond interface and associate the file in the Synthesis strategy settings as prescribed. This ensures that source and .fdc constraints files are recognized and associated with your Incremental Design flow project in Diamond.

Preference/net naming issues Preference and net names can vary from design to design in the Incremental Design flow for several possible reasons.

Issue: In some cases preferences that are honored in the normal flow are not honored in the incremental flow. For example, a clock name has significantly changed which leads to a FREQUENCY constraint, related to that clock not honored; a net name has changed which leads to a MAXDELAY constraint, related to that net, not honored; or a Dynamic Clock Select (DCS) name has changed which leads to USE PRIMARY DCS NET "<net_name>" "<quadrant_location>" not honored. This may be due to synthesis optimization being performed differently with compile points present as opposed to when they are absent from a design.

Solution: When using incremental design, you should double check Map warnings about preferences. If you see warnings indicating that preferences are ignored which are accepted in normal flow, you may need to modify your preference file, or use synthesis attribute “syn_keep” to preserve net names. Another method, for checking changed clock names, is to open Spreadsheet View after running the Translate Design process in Diamond. In the Spreadsheet View, check the Clock Resource tab. This tab lists all of the clock names that are recognized by the tool, including those that have their names changed. Users should use the names listed in this tab to set their constraints. As always, it is better to constrain ports rather than nets if possible, as they will not be renamed.

UGROUPs in guided iterations In certain cases UGROUPs you assign to unchanged SLICE components as attributes or preferences are not honored.

Issue: When you are running in the guided iteration run flow mode and you attempt to move unchanged SLICEs relative to the reference design by adding a UGROUP to HDL or preference file, the placer will not, by default, honor the new UGROUP location. There is no error or warning messages to alert you to this condition.

Solution: To avoid this situation, have your UGROUPs defined where you desire them to be located when you run the first implementation. While changed logic can be located via a UGROUP, this will only work where there are empty sites available. The placer will not rip up any unchanged placed logic from a reference design by default.

Tcl Commands Can Only Be Run Through Tcl Shell The database extended Tcl commands and Partition Extended Tcl commands can only be run through a Tcl Shell, and cannot be run in the Diamond GUI.

General Warnings & Errors You may encounter the following general warnings and errors when using the Incremental Design flow.

- If the user's command has a reference file specified, such as par -w ... <*file_name*>.ncd, but file.ncd does not exist, PAR will switch to Unguided iteration with the following warning message:

  ```
  Failed to open reference file %s, switch to unguided
  iteration
  ```

- If the user's .icf file has no partition, or all partitions are disabled, PAR will switch to normal flow with the following error message:

  ```
  No partition is defined or all partitions are disabled,
  switch to normal flow.
  ```

- If the user's command + d has a reference file specified, such as par -w ... -<*file_name*>.ncd, and the level and effort settings in the Partition Manager meet one or both of the following conditions:

  - Preservation data = synthesis and any reimplementation effort.
  - Preservation data = Map, reimplementation effort = unguided.

  PAR will switch to Unguided iteration with the following warning message:

  ```
  Because of the level/effort setting in icf file, redo
  unguided iteration.
  ```

Note

If this warning occurs, the Floorplan view will be invoked, so the regions of partitions will be changed.

- If the user's command has a reference file specified, such as par -w ... -<*file_name*>.ncd, but one of partitions has changed the anchor or bbox through the Partition Manager, PAR will switch to unguided iteration with the following warning message:

  ```
  Because some of the partitions changed partition_plan, guide
  iteration is disabled; The regions in icf files will be
  honored but the locations of comp will be changed.
  ```

Map Warnings & Errors You can encounter the following warning and errors in the Incremental design flow when using the Map process.

```
WARNING - map: There is no partition, and the option "-inc" is
ignored.
```

This condition may be the result of the following:

- No partition has been found so Map will run in normal mode.
- No ICF file will be created.
- The “par -inc -icf” command may fail due to missing ICF file.
- There is no partition, so you cannot use the incremental flow.

```
WARNING – map: The time stamp %s for partition %s in current
design is older than the reference one %s.
```

This condition may be the result of the following:

- The time stamp for a given partition in current design is older than the reference one. This condition may have occurred when a modified design is used as a reference for the previous design. This practice is not recommended in the Incremental Design flow.
- The partition will be marked as “CHANGED”.

```
ERROR – map: Sorry, there is no license for %s.
```

This condition may be the result of the following:

- The Incremental Design flow requires a special feature license. If you do not have it in your license file, you will see the message. Go to the Lattice web site and contact Sales support to obtain a license.

```
ERROR – map: Sorry, device %s is not supported for %s.
```

This condition may be the result of the following:

- Incremental flow may not be supported for all Lattice FPGA/CPLD families. You get this message because the selected device is not supported for incremental flow.

Placer Warnings & Errors You can encounter the following warning and errors in the Incremental design flow as a result of the placer processing your design.

- For running in a multi-par (multiple seeds) format, after each seed, par may issue one of the following three kind of messages during the Incremental Design flow:

  `WARNING: “message …”`

  PAR continues after this warning message is issued.

  `ERROR: “message …”`

  PAR will exit after it issues this error message.

  ```
  WARNING: placement not success in this seed and will
  continue next seed if exists.
  “message …”,
  ```

PAR will abort running for the current seed after this warning message and try the next seed. The message could be one of the follows:

```
ERROR :
"Internal database corrupted:  NULL site in slice
placement"

NOT SUCCESS :
"Fatal Error in preferplace(): cannot find initial
placement for UPGROUPs. Please check the UPGROUP
preferences to alleviate constraints."

"Impossible to place SLICE_10 in UPGROUP GROUP1 in prefer
location."

"Placer could not relatively place comp SLICE_10 at
R15C8D(x=61/y=8."

"Failed to find the anchored location."
```

- Anchoring error during PGROUP pre-placement:

```
pgroup=SLICE_10 anchor=R15C8D. Not enough available sites
in bbox for all group comps. Please check for multiple
LOCATE/REGION/QUADRANT design preferences leading to
highly constrained PGROUP placement regions.

Error in pgroup initial placement!

Comp 'SLICE_10' remains unplaced after Phase 0
flattening.

set_initial_anchors failure on PGROUP placement"

WARNING:

switch to high-effort for PGROUP initial placement due to
congestion...
```

The following are messages issued by the placer. For each PAR incremental design run, the PAR file will have the following info:

```
Starting incremental place and route ...
```

Incremental design will exit if you encounter any of the following PAR error messages:

```
ERROR: No *.icf file for incremental par flow.
ERROR: Failed to open file design.icf for incremental par flow.
ERROR: Incremental flow can't support guide file with option "-
g filename".
ERROR: option -g and -ref can't be appeared in command line
simultaneously.
ERROR: file name missing after "-ref".
ERROR: Failed to open reference file design_ref.ncd.
```

ICF Parser Warnings & Errors The following are messages issued by the ICF parser. If the ICF file was loaded successfully, it will issue the following message:

```
Completed loading 3 partition data.
```

- If the ICF parser detects an error, it may issue any of the following error messages listed below:

```
ERROR: comp%d %s belong to both partition %s and %s.
ERROR: Illegal keyword XXX
ERROR: Must assign value for keyword XXX for partition P0.
ERROR: Invalid icf string
ERROR: Fail to open ICF file XXX.
ERROR: No statement in ICF file XXX.
ERROR: Illegal icf key-value XXX.
ERROR: XXX YYY is an illegal pair , value must be in "ZZZ".
```

Then PAR will exit after printing out the following line:

```
There are %d errors in icf file.
```

Router Warnings & Errors You can encounter the following warning and errors in the Incremental design flow as a result of the router processing your design.

- Usually, in incremental flow, the positions of primary clock signals are copied according to the reference design. However, for any reason, if the position of a primary clock signal has been preassigned, the copy will be skipped and you will encounter the following warning message:

```
WARNING – par: The pclk position for the signal %s has been
already assigned, and the assignment from the guided/
reference design is ignored.
```

- As mentioned in the prior case, in incremental flow, the clock assignment is usually copied from reference design. However, for any reason, if the target position has been used by other signals, the copy will be skipped and you will encounter the following message:

```
WARNING – par: The pclk position %d has been already
occupied. The pclk assignment for the signal %s copied from
guided/reference design is ignored.
```

- Usually in the incremental flow, the positions of secondary clock signals are copied according to the reference design. However, for any reason, if the position of a secondary clock signal has been preassigned, the copy will be skipped and you will encounter the following message:

```
WARNING – par: The sclk position for the signal %s has been
already assigned, and the assignment from the guided/
reference design is ignored.
```

- As mentioned above, in incremental flow, the clock assignment is usually copied from reference design. However, for any reason, if the target position has been used by other signals, the copy will be skipped and you will encounter the following message:

```
WARNING – par: The sclk position %d has been already
occupied. The sclk assignment for the signal %s copied from
guided/reference design is ignored.
```

Supplying Archives for Technical Support As a general advisory, if you encounter issues that require debugging by Lattice technical support, it would be extremely helpful to provide project archives both for the initial reference design files and the guided iteration run flow mode version where the issue was encountered.

As described earlier in this guide in the section, “Archiving Incremental Design Results” on page 662, make sure that at the appropriate points in the flow that you archive your project.

Chapter 9

Analyzing Static Timing

Static timing analysis (STA) is a method for determining if your circuit design meets timing constraints. It is a method that does not require the use of simulation. The STA process employs conservative modeling of gate and interconnect delays that reflect different ranges of operating conditions on various dies, providing complete verification coverage.

This section explains how to use Diamond's static timing analysis tools and reports to help meet your design's timing constraints.

See Also ▶"Static Timing Analysis Tools" on page 694

- "Running TRACE" on page 696
- "Using Timing Analysis View" on page 732

Static Timing Analysis Tools

Diamond provides two primary tools for performing static timing analysis on timing preferences: the timing reporter and circuit evaluator (TRACE) and the Timing Analysis view.

Additionally, the I/O Timing Analysis process, which is available after Place & Route, generates an I/O Timing Report (.ior) that is not limited to timing preferences.

TRACE TRACE analyzes timing preferences that are present in the logical preference (.lpf) file. These timing preferences are defined in the Timing Preferences sheet of Spreadsheet View or in a text editor before the design is mapped. A TRACE report file, which shows the results of timing preferences, is generated each time you run the **Map Trace** process or the **Place & Route Trace** (PAR) process. The results can then be viewed in the Report View window. The Map TRACE report (.tw1) contains estimated routing that can be used to verify the expected paths and to provide an estimate of the delays before you run Place & Route. The PAR TRACE report (.twr) contains delays based on the actual placement and routing and is a more realistic estimate of the actual timing. See "Running TRACE" on page 696.

Timing Analysis View The Timing Analysis view is a graphical view of the post-route TRACE report. It provides path tables, schematic views of timing paths, and a report of each timing preference. It also allows you to cross-

probe to Floorplan View or Physical View to see where these paths exist on the chip. See "Using Timing Analysis View" on page 732 and "Cross-Probing from Timing Analysis View" on page 753.

Timing Analysis view also includes a simplified version of Spreadsheet View that enables you to create flow-independent timing preference files (.tpf) for a given implementation. With the .tpf file input, the Timing Analysis view computes the results of your hypothetical timing constraints and allows you to discover how they might affect your design. You can later copy revised timing preferences from the .tpf file to the logical preference file. See "Creating a New Timing Preference File" on page 747 and "Modifying Preferences for Timing Analysis" on page 734.

I/O Timing Analysis The I/O Timing Analysis process, which is available after placement and routing, generates an I/O Timing Analysis Report (.ior). Unlike TRACE, the I/O analysis is not limited to defined timing preferences. Paths that have not been constrained with timing preferences will still be reported. Also, the verbose version of the I/O timing analysis sweeps across multiple performance grades to find the worst case results. See "Running I/O Timing Analysis" on page 722.

See Also
- "Running TRACE from the Command Line" on page 2526
- "Running I/O Timing from the Command Line" on page 2531
- "Preferences to Improve Timing" on page 380
- "Setting Timing Preferences" on page 492

Strategies for Timing Analysis

Reporting options for TRACE and I/O timing analysis are defined using strategies. Strategies are located in the Strategies folder of the File List view. You can associate a strategy with an active project implementation by right-clicking a strategy and choosing **Set As Active Strategy** from the pop-up menu.

By default, the Timing Analysis view will use your active strategy as its default TRACE settings when you initially open the view. Your Map and PAR Worst Case Paths setting defaults to reporting 10 in the TRACE report; however, Diamond will automatically run 30 to cache timing data in memory so that performance in the Timing Analysis view is not diminished.

For more general information on setting strategies in Diamond, see "Using Strategies" on page 48.

See Also
- "Using Timing Analysis View" on page 732
- "Setting Options in Timing Analysis View" on page 733

Running TRACE

You can access the TRACE program from the Diamond Process window for Map Design and Place & Route Design. You can also access it using the `trce` command in the Tcl Console, a DOS console, or from an xterm in Linux. TRACE runs analysis on timing preferences that have been specified in the logical preference file (.lpf) or the HDL source, and it generates a report of the results. You can view the TRACE report in the Reports window.

To run TRACE in the Diamond environment:

- In the Process view, double click **Map Trace** or **Place & Route Trace**.

  TRACE runs static timing analysis on your design and generates a map TRACE report (.tw1) or a PAR TRACE report (.twr).

To run TRACE from the command line:

- Type **`trce`** on the command line with, at minimum, the names of your input .ncd and .prf files. For example,

  ```
  trce <file_name>.ncd <file_name>.prf
  ```

For more information, see "Running TRACE from the Command Line" on page 2526.

See Also "Static Timing Analysis Tools" on page 694

- "Using Timing Analysis View" on page 732

Setting TRACE Options

TRACE options are available in the active Strategy for an implementation. The options provide you with control over how timing data is reported.

To set TRACE options in the Diamond environment:

- Choose **Project > Active Strategy**, and then choose either **Map Trace Settings** or **Place & Route Trace Settings** from the drop-down menu.

  Optionally, double click the active Strategy in the Strategies folder of File List view, and then select **Map Trace** or **Place & Route Trace**.

The following STA options are available for both Map and Place & Route TRACE:

- Analysis Options
  - Standard Setup and Hold Analysis (default) – Performs both the Standard Setup Analysis and the Hold Analysis.
  - Hold Analysis – Performs hold analysis.
  - Standard Setup Analysis – Performs setup time checks on applicable preferences. For post-route analysis, you can override the default

performance grade (the performance grade of the target device) by using the Speed for Setup Analysis option.

- Standard Setup with Hold Analysis on IOs – Performs standard setup analysis and a hold time analysis on applicable I/O preferences in the same report file. For post-route analysis, you can override the default performance grade (the performance grade of the target device) by using the Speed for Setup Analysis option.

- Auto Timing – lists the auto-generated timing preferences in the TRACE report.

  If you do not define any timing preference for your FPGA design, Diamond will automatically generate timing preferences for you. Set this to True (default) if you want your TRACE report to list the auto-generated timing preferences.

- Check Unconstrained Connections – lists the connections that are not covered by any timing preference.

  Note

  The Check Unconstrained Connections option will be discontinued after the next two Diamond releases.

- Check Unconstrained Paths – reports the paths that are not constrained and shows the start point and end point of each path. TRACE will suggest some timing preferences to constrain the given paths. The unconstrained paths are shown only in the “setup” timing check report to avoid duplication of these same paths in the “hold” timing check report.

  Based on the design and the required performance, only necessary paths should be constrained so that PAR focuses only on the optimization of the important paths. However, the Unconstrained Paths section of the TRACE report is very useful for identifying whether any missing timing constraints are really important to the design. This option does not require you to add more preferences in an attempt to constrain all paths. Instead, it serves as a reminder that there might be a necessary preference that is missing, which could impact the desired performance of the design.

  Important points about connection coverage:

  - Some signal connections might not be constrained by any of the four predefined types of preferences. It is possible that all pairs of source and sink are covered by the given preferences, but there could still be connections not covered by any of the given preferences. Therefore, the connection coverage might be less than 100% even though no unconstrained path of the four predefined preference types is found.

  - Sometimes an unconstrained path is found even though the connection coverage is reported as 100%. The unconstrained path is about the path between a source and sink pair. If a legal source and sink pair is not covered by a preference, it will be reported as an unconstrained path. But because the connections on this

unconstrained path could already be covered by some other preferences, the connection coverage could still be 100%.

Note

The Check Unconstrained Paths option cannot be used with the -allprefpath command-line option.

- Full Name – when set to True, the TRACE report will show full-length component names instead of the truncated names.

  This option defaults to having truncated COMP names appear in the TRACE report (.tw1 or .twr) file. COMP names are truncated to a certain character limit and preceded by an asterisk. The expanded option allows the entire name to be reported; however, longer COMP names can cause some tabular formats in the ASCII file report sections to become misaligned, making the report less readable.

- Number of Unconstrained Paths (0 to 4096) – controls the number of unconstrained paths that are reported. When set to 0, all unconstrained paths are reported.

Note

If you enter a value of 500 or more for the number of paths to be reported, you might experience problems related to excessive file size.

If the -allprefpath option is used in the command line, it will override the unconstrained paths option.

- Report Asynchronous Timing Loops – produces a report of paths that could not be analyzed because they are asynchronous loops.
- Report Style – sets the format of the TRACE report.
  - Verbose Timing Report (default) – lists detailed logic and route delays for all constrained paths and nets in the design.
  - Error Timing Report – lists logic and route delays for each path and net that causes a timing error.
- Route Estimation Algorithm (%) – configures the route estimation algorithm for pre-route static timing analysis.

  0 (default) – Enables route delay estimation based on a suite of Lattice benchmark designs. Note that delay estimates are based on the default performance grade of the target device.

  The default 0 value will run an algorithm that considers only the most critical paths of each design in a suite of designs. For each critical path, an average route delay is determined for each logic-logic connection. This is further decomposed to the number of switches per connection, and delay per switch. Each device family and performance grade will have its own unique value(s). Special cases that deviate significantly from the general routing scenario are excluded in the calculation and also handled separately by PAR. For example, carry chains are not included in the average route delay value, and the average route delay value is not used by MAP to estimate the delay in a carry chain.

1~100 – specifies logic delay as a percentage of the overall path delay where the total delay is the sum of logic and route delays.

The % Logic Delay option is a simple method that correlates the logic delay to the routing delay. If a 50/50 correlation is chosen (50 percent), then the routing delay of a path is assumed to be the same as the logic delay of that path. However, there is not necessarily any correlation between a path's logic delay and its routing delay; for example, an adder's carry chain has very low routing delay versus logic delay. So this method does not lead to accurate estimates, but it can still be used for such tasks as checking the number of logic levels in the design and syntax checking the preferences.

- Worst Case Paths (0-4096) – specifies the number of paths to be reported for each timing preference. Enter zero or a greater value. The default value is 10; however, Diamond will automatically run 30 to cache timing data in memory so that performance in the Timing Analysis view is not diminished. You will see additional data in your reports for this reason.

Note

If you enter a value of 500 or more for the number of paths to be reported, you might experience problems related to excessive file size.

When Verbose Timing Report is selected, the value you specify limits the number of paths reported for each timing preference (0 = no limit). There is no practical limit, although unlimited reporting can exhaust available memory and disk space.

Scoring a path refers to the timing extraction performed by the static timing analysis engine. TRACE will always extract the worst case paths, but you can choose to analyze and score more than the system default.

Given a large design with many paths to examine, you might wish to limit the report size by using more specific preferences or BLOCK type exceptions temporarily to mask paths that are not of interest.

When Error Timing Report is selected, the specified number of paths with timing error will be printed out. If you enter 0, no path will be reported.

- Speed for Hold Analysis – allows you to override the default -M (minimum) performance grade for hold time analysis, which represents faster silicon than the fastest performance grade of the device being targeted.
- Speed for Setup Analysis – allows you to override the default performance grade which runs setup analysis against the performance grade of the device currently targeted by the project implementation.

See Also

- "Static Timing Analysis Tools" on page 694
- "Running TRACE" on page 696
- "Using Timing Analysis View" on page 732
- "Setting Options in Timing Analysis View" on page 733

Viewing TRACE Reports

TRACE reports are outputs of the Map Trace and Place & Route Trace processes. After running either of these processes in Diamond's Process view, you can examine the associated TRACE reports in the Reports window. The Map TRACE report (.tw1) and the Place & Route TRACE report (.twr) files are written out to your project directory in ASCII format using the current project name for the file name.

To view the TRACE report file in the Report window:

1. Click the **Reports** tab to activate the Report window.
2. In the Design Summary pane, select **Map Trace** or **Place & Route Trace** from the Analysis Reports folder.

   The selected report appears in the Reports window. The Design Summary pane presents a hierarchy view of the report.

   - Expand a section of the report in the Design Summary pane and click a subsection name to open the section in the Reports window.
   - In the Report window, right-click and choose Find in Text to search for a specific preference or a design element.

To view the TRACE report file in a text editor:

- Open your text editor and navigate to the project directory. The TRACE reports are usually located in the <project_name><project_name> folder.
  - To open the Map TRACE report, select the <project_name>.**tw1** file.
  - To open the Place & Route TRACE report, select the <project_name>.**twr** file.

See Also "TRACE Report File" on page 700

- "Running TRACE" on page 696
- "Setting TRACE Options" on page 696
- "Setting Options in Timing Analysis View" on page 733

TRACE Report File

The TRACE Report File (.tw1/.twr) is an ASCII report that enables you to determine to what extent the timing constraints for a design have been met. The .tw1 TRACE report output file is a result of the Map Trace process in Diamond. The .twr TRACE report output file is a result of the Place & Route Trace process.

Two levels of reporting are available for TRACE reports: verbose and error. By default, Diamond generates a verbose timing report. If desired, you can change this setting in the active strategy dialog box so that an error timing report is generated.

Facts regarding Error and Verbose levels of TRACE timing reporting:

- Both report styles conclude with Timing Summary subsections.
- The number of items reported can be limited for each preference in either error or verbose mode.
- The TRACE output format supports logical names.
- The error report lists timing errors and associated net/path delay information.
- The error and verbose reports also contain an asynchronous loop report that shows all paths that cannot be analyzed.
- The verbose report lists delay information for all nets and paths.
- For both types of reports, if you specify a preference file that contains invalid data, a list of preference file errors appears at the beginning of the report.
- If preferences for process, temperature, or voltage exist in a preference file, these preferences are included in the timing report.
- In both error and verbose reports, the details of the paths that contribute to clock skew are reported.
- The result of maximum frequency (f_{MAX}) in TRACE is potentially the highest frequency that the design can achieve. However, there are factors that can change the way in which f_{MAX} is reported. For example, if the f_{MAX} path is PLL-relevant and the PLL is configured as cycle "SHIFT", then the maximum frequency will change PLL "SHIFT" delay accordingly, which might impact clock skew and f_{MAX} results. Final f_{MAX} results depend upon how PLL is used in each individual case.

Note

TRACE supports the preferences "CLKSKEWDIFF" on page 1206 and "CLKSKEWDISABLE" on page 1207, which allow you to control clock skew computation.

TRACE Error Report

The main body of the error report lists all timing preferences as they appear in the input preference file. For each preference that is met, the report simply states the number of items scored by TRACE, reports no timing errors detected, and issues a brief report line that indicates important information such as the maximum delay for the particular preference.

For each preference that is not met, the report shows the number of items scored by TRACE, the number of errors encountered, and a detailed breakdown of the error. In addition, TRACE reports clock skew details for placed and routed designs.

A Timing Summary always appears at the end of the report. The Timing Summary will include any preferences that are not met and mark them with an asterisk.

The Logical Details section, which reports circuit path source and destination information, appears at the beginning. If a path begins or ends at a sequential element, such as a flip-flop, the clock signal associated with the flip-flop is reported in the path delay report. For example:

```
10.150ns delay CLKD1 to CLKD1' exceeds
 10.000ns delay constraint less
  3.000ns setup requirement (totaling 7.000ns) by 3.150ns
  Delay   Site    Resource
 3.000ns CLB_R2C1.K to CLB_R2C1.XQ CLKD1 (from CLK)
 1.310ns CLB_R2C1.XQ to CLB_R2C1.G2 CLKD1
 4.500ns CLB_R2C1.G2 to CLB_R2C1.Y CLKD1
 1.340ns CLB_R2C1.Y to CLB_R2C2.C1 CLKD1D2 (to CLK)
 -------
 10.150ns (73.9% logic, 26.1% route), 2 logic levels.
```

Source and destination clock signals are listed because the path originates at a flip-flop (CLKD1.K, signal CLK) and terminates at a flip-flop with a 3 ns setup requirement relative to the signal CLK. This allows you to determine what the source and destination clock signals are for sequential paths.

The following example has source and destination clock signals that are different. Identifying these clocks allows you to determine whether or not the timing constraint on the path needs modification based on the phase relationships of the different clock signals.

```
4.830ns delay CLKD2 to CLKD2' meets
 20000ns delay constraint less
 3.000ns setup requirement (totaling 17.000ns) by 12.170ns
  Delays   Site     Resource
 3.000ns CLB_R3C1.K to CLB_R3C1.XQ CLKD2 (from CLK2)
 1.830ns CLB_R3C1.XQ to CLB_R3C2.C1 CLKD2 (to CLK)
 -------
 4.830ns (62.1% logic, 37.9% route), 1 logic levels.
```

Verbose TRACE Report

The verbose report is similar to the error report but provides more details on delays for all constrained paths and nets in the design. As in the other two types of reports, descriptive material appears at the top.

The body of the verbose report enumerates each preference as it appears in the input preference file, the number of items scored by TRACE for that preference, and the number and description of errors detected for the preference. A report line for each preference provides important information, such as the amount of delay on a net and by how much the constraint is met.

For path preferences, if there is an error, the report indicates the amount by which the preference is exceeded. If there are no errors, the report indicates that the preference passed and by how much. Each logic and route delay is analyzed, totaled, and reported.

The TRACE report includes all paths that are covered with the BLOCK PATH preference. Details for a BLOCK NET preference are not included. In TRACE, the signal defined in a BLOCK NET preference is removed from the timing-graph analysis completely. Because there is no path or coverage related to this signal any longer, all paths through the signal are blocked/removed and considered covered.

TRACE Report Sections

Shown below are the possible sections that a given TRACE Report (.tw1/.twr) file might include. These sections will vary, depending on the FPGA architecture, timing preferences, design elements, and reporting style (error or verbose).

Report Title and Design Information At the top of each type of report is descriptive information about your design. For example, it includes the software version of the application, the input preference file name, the NCD design file name, device specifications (device, package, performance grade), and environment path name. The example below identifies this as a PAR TRACE report instead of a Map TRACE Report.

```
Place & Route TRACE Report

Loading design for application trce from file
attributes_attributes.ncd.
Design name: top
NCD version: 3.2
Vendor:      LATTICE
Device:      LFXP2-17E
Package:     FTBGA256
Performance: 5
Loading device for application trce from file
'mg5a50x47.nph' in environment: C:/lscc/diamond/2.1/
ispfpga.
Package Status:                     Final
Version 1.63
Performance Hardware Data Status:   Final
Version 10.6
Setup and Hold Report
```

Version, Build, Time Stamp, and Copyright This section shows the date and time the file was generated and the software version that created it. All relevant copyright information is also displayed.

Report Information This section tells you about the report file that TRACE generated, the exact command that generated it, the preference file (PRF) and design NCD file associated with it, and the reporting level of the file.

Preference Summary The Preference Summary lists all the timing preferences that you specified in the LPF file. It shows the actual preference line syntax, the number of errors, and the number of items scored by TRACE.

```
Preference Summary

FREQUENCY NET "dec_pll_clk_c" 200.000000 MHz (28 errors)

            511 items scored, 28 timing errors detected.
Warning: 184.298MHz is the maximum frequency for this
preference.

FREQUENCY NET "enc_pll_clk_c" 200.000000 MHz (4 errors)

            101 items scored, 4 timing errors detected.
Warning: 196.232MHz is the maximum frequency for this
preference.

FREQUENCY NET "dec_clk_c" 200.000000 MHz (0 errors)
            0 items scored, 0 timing errors detected.

FREQUENCY NET "enc_clk_c" 200.000000 MHz (0 errors)
            0 items scored, 0 timing errors detected.

BLOCK PATH TO CELL "u1_decoder/cnt_enb" (0 errors)
            112 items scored, 0 timing errors detected.

BLOCK ASYNCPATHS
BLOCK RESETPATHS
```

Preference Details The Preference Details section shows the logical details and delays of each scored path covered by a given timing constraint.

It provides timing error details or warnings on a constraint requirement for physical path delay. It also shows attributes associated with I/O logic, interface timing AIL, and DDR input dynamic timing settings. The boldfaced line in the following example shows where the 2.528ns DIN_SET delay originates:

```
================================================================
Preference: FREQUENCY PORT "CLK" 1000.000000 MHz ;
            1 item scored, 1 timing error detected.
----------------------------------------------------------------

Error:  The following path exceeds requirements by 2.272ns

 Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

   Source:         Port       Pad            d
   Destination:    FF         Data in        Q_0io  (to CLK_c +)

   Delay:               0.744ns  (100.0% logic, 0.0% route), 1 logic levels.

 Constraint Details:

      0.744ns physical path delay d to d_IOLOGIC_S exceeds
      1.000ns delay constraint less
      2.528ns DIN_SET requirement (totaling -1.528ns) by 2.272ns

IOLOGICB20A attributes: COARSE=CDEL1, FINE=FDEL10

 Physical Path Details:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     0.744         R9.PAD to       R9.PADDI d
ROUTE         1     0.000       R9.PADDI to IOLOGICB20A.DI d_c (to CLK_c)
                  --------
                    0.744   (100.0% logic, 0.0% route), 1 logic levels.

Warning: 305.623MHz is the maximum frequency for this preference.
```

The Logical Details subsection reports circuit path Source and Destination information:

- instance names as they appear in the EDIF file for the following cell types: Registered elements, Ports, RAM instances
- a description of the type of function performed at the destination; for example, Data in, Write enable, or Carry in
- indication of the triggering edge of the clock using (+/-)

The section includes source component logical name, cell type, pin type, clock name and edge; and it includes destination component logical name, cell type, pin type, clock name and edge.

The Physical Path Details shows the paths connected to their source and destination sites on the chip. At the bottom of the Delays column, it shows the total amount of delay for the path, the percentages of the total allocated to logic and to route delays (in parentheses), and the number of logic levels (components) involved in the preference.

This part of the TRACE report will also alert you to any timing warnings or errors associated with the specific preference. Timing warnings point out potential timing problems. Timing errors, however, indicate absolute timing constraint violations. Timing errors may indicate a need for design modifications and/or multiple placement and/or reentrant routing.

The most critical paths are reported from top to bottom in this section. These paths are all ranked in terms of weighted slack values based on one full clock cycle. If the path is not a full cycle, you might see something like “weighted slack = 2.000ns” following the pass/fail statement. This simply indicates that the actual timing value is scaled up to a full clock cycle value and that the path was scored at 1.00ns at a half cycle.

To discover the multiplier that was used to scale the clock time to one full cycle, look at the source and destination clocks in the Logical Details section of a given path. For example, if the clock frequency is the same and you go from a positive to positive edge clock or negative to negative, you have a full cycle. If your clocks with the same frequency go from a positive to a negative or vice versa, that will be a half cycle. If you have clocks with different frequencies, interdomain clocks, your cycle time will vary; the multiplier will not be readily apparent, but it will be scaled up appropriately with a weighted slack value.

```
================================================================================
Preference: FREQUENCY PORT "clk" 150.000000 MHz ;
            6 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------


Passed:  The following path meets requirements by 0.235ns

 Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

   Source:         FF         Q              c_0  (from clk_c +)
   Destination:    FF         Data in        c_0  (to clk_c +)

   Delay:               0.236ns  (75.8% logic, 24.2% route), 2 logic levels.

 Constraint Details:

       0.236ns physical path delay SLICE_0 to SLICE_0 meets
       0.001ns DIN_HLD and
       0.000ns delay constraint less
       0.000ns skew requirement (totaling 0.001ns) by 0.235ns

 Physical Path Details:

   Name    Fanout   Delay (ns)          Site               Resource
REG_DEL     ---     0.120     R25C7B.CLK to      R25C7B.Q0 SLICE_0 (from clk_c)
ROUTE         4     0.057      R25C7B.Q0 to      R25C7B.D0 c_c_0
CTOF_DEL    ---     0.059      R25C7B.D0 to      R25C7B.F0 SLICE_0
ROUTE         1     0.000      R25C7B.F0 to     R25C7B.DI0 c_c_i_0 (to clk_c)
                  --------
                    0.236   (75.8% logic, 24.2% route), 2 logic levels.

 Clock Skew Details:

 Source Clock:
           Delay              Connection
          0.317ns         P4.PADDI to R25C7B.CLK

 Destination Clock:
           Delay              Connection
          0.317ns         P4.PADDI to R25C7B.CLK
================================================================================
Preference: INPUT_SETUP ALLPORTS INPUT_DELAY 5.000000 ns HOLD 0.000000 ns CLKPORT
"clk" ;
            0 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------
```

The details on the following MAXDELAY preference are followed with warning and error messages:

```
================================================================================
Preference: MAXDELAY NET "*sw{1:6}_smii_rx_clk*" 1.000000 nS ;
            6 items scored, 5 timing errors detected.
--------------------------------------------------------------------------------

Error:     2.034ns delay on sw3_smii_rx_clk_c exceeds
           1.000ns delay constraint by 1.034ns

           Delays             Connection(s)
           0.643ns         J2.PADDI to IOL_L23B.CLK
           0.732ns         J2.PADDI to IOL_L26A.CLK
           0.916ns         J2.PADDI to IOL_L28A.CLK
           0.659ns         J2.PADDI to IOL_L25A.CLK
           0.732ns         J2.PADDI to IOL_L26B.CLK
           0.916ns         J2.PADDI to IOL_L28B.CLK
           0.659ns         J2.PADDI to IOL_L24A.CLK
           0.659ns         J2.PADDI to IOL_L24B.CLK
           0.659ns         J2.PADDI to IOL_L25B.CLK
           1.445ns         J2.PADDI to R14C20B.CLK
           1.445ns         J2.PADDI to R14C20C.CLK
           2.034ns         J2.PADDI to R13C34C.CLK
           1.617ns         J2.PADDI to R13C20A.CLK
           1.246ns         J2.PADDI to R15C16B.CLK
           1.434ns         J2.PADDI to R15C17B.CLK
           1.434ns         J2.PADDI to R15C18B.CLK

Warning:   2.034ns is the maximum delay for this preference.
```

When the design contains a pulse width error—for example, when a FREQUENCY preference that specifies an unachievable clock speed on a DSP in your design is not met—TRACE reports these conditions as shown below:

```
============================================================
Preference: FREQUENCY PORT "clkfast" 300.000000 MHz ;
            169 items scored, 1 timing errors detected.
Note: Component internal maximum frequency has been exceeded.
------------------------------------------------------------
```

Note

These examples show an input design that is placed and routed. If the input design is neither placed nor routed, estimated delays will all show up as 0 and will be signified by the letter **e**.

Unconstrained Paths This section will be included in the report if you have selected the Check Unconstrained Paths Trace option for the active strategy. In the Unconstrained Paths section, TRACE reports the paths that are not constrained and shows the start point and end point of each path. Suggested timing preferences are provided to constrain the given paths.

Based on the design and the required performance, only necessary paths should be constrained so that PAR focuses only on the optimization of the important paths. However, the Unconstrained Paths section of the TRACE report is very useful for identifying whether any missing timing constraints are really important to the design. This option does not require you to add more preferences in an attempt to constrain all paths. Instead, it serves as a reminder that there might be a necessary preference that is missing, which could impact the desired performance of the design. See "Setting TRACE Options" on page 696 for important points about connection coverage.

As shown in the following example, TRACE reports only the "setup" timing of unconstrained paths to avoid duplication of these same paths in a "hold" timing report.

```
Preference: Unconstrained Paths
            101 items scored, 0 timing errors detected.
--------------------------------------------------------------
Unconstrained Preference:
   INPUT_SETUP PORT "tx_dword_7" CLKNET "enc_pll_clk_c"

Report:    5.483ns delay tx_dword_7 to u1_encoder/SLICE_47 (5.390ns delay and 0.093ns
setup)

   Name    Fanout   Delay (ns)          Site               Resource
PADI_DEL    ---     1.049        A15.PAD to      A15.PADDI tx_dword_7
ROUTE         2     2.233      A15.PADDI to     R18C46D.D1 tx_dword_c_7
CTOF_DEL    ---     0.260     R18C46D.D1 to     R18C46D.F1 u1_encoder/SLICE_69
ROUTE         1     0.629     R18C46D.F1 to     R17C46B.D1 u1_encoder/parity_8
CTOF_DEL    ---     0.260     R17C46B.D1 to     R17C46B.F1 u1_encoder/SLICE_47
ROUTE         1     0.699     R17C46B.F1 to     R17C46B.B0 u1_encoder/parity
CTOF_DEL    ---     0.260     R17C46B.B0 to     R17C46B.F0 u1_encoder/SLICE_47
ROUTE         1     0.000     R17C46B.F0 to    R17C46B.DI0 u1_encoder/data_reg_3_16
(to enc_pll_clk_c)
                  --------
                    5.390   (33.9% logic, 66.1% route), 4 logic levels.

Unconstrained Preference:
   INPUT_SETUP PORT "tx_dword_0" CLKNET "enc_pll_clk_c"

Report:    5.279ns delay tx_dword_0 to u1_encoder/SLICE_47 (5.186ns delay and
0.093ns setup)

   Name    Fanout   Delay (ns)          Site               Resource
PADI_DEL    ---     1.049        F11.PAD to      F11.PADDI tx_dword_0
ROUTE         2     1.608      F11.PADDI to     R18C46D.C0 tx_dword_c_0
CTOF_DEL    ---     0.260     R18C46D.C0 to     R18C46D.F0 u1_encoder/SLICE_69
ROUTE         1     1.050     R18C46D.F0 to     R17C46B.B1 u1_encoder/parity_9
CTOF_DEL    ---     0.260     R17C46B.B1 to     R17C46B.F1 u1_encoder/SLICE_47
ROUTE         1     0.699     R17C46B.F1 to     R17C46B.B0 u1_encoder/parity
CTOF_DEL    ---     0.260     R17C46B.B0 to     R17C46B.F0 u1_encoder/SLICE_47
ROUTE         1     0.000     R17C46B.F0 to    R17C46B.DI0 u1_encoder/data_reg_3_16
(to enc_pll_clk_c)
                  --------
                    5.186   (35.3% logic, 64.7% route), 4 logic levels.
```

Blocked Path The TRACE report includes all paths that are covered with the BLOCK PATH preference, as shown in the following example.

```
================================================================================
Preference: BLOCK PATH TO CELL "u1_decoder/cnt_enb" ;
            112 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------


Blocked:

 Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

   Source:         FF         Q              u1_decoder/sync_sftreg[16]  (from
dec_pll_clk_c +)
   Destination:    FF         Data in        u1_decoder/cnt_enb  (to dec_pll_clk_c +)

   Delay:               7.843ns  (21.5% logic, 78.5% route), 6 logic levels.

   Name    Fanout   Delay (ns)          Site               Resource
REG_DEL     ---     0.383    R30C29A.CLK to     R30C29A.Q0 u1_decoder/SLICE_31 (from
dec_pll_clk_c)
ROUTE         9     1.541     R30C29A.Q0 to     R33C30A.A0 u1_decoder/sync_sftreg[16]
CTOF_DEL    ---     0.260     R33C30A.A0 to     R33C30A.F0 u1_decoder/SLICE_75
ROUTE         1     0.278     R33C30A.F0 to     R33C30A.D1 u1_decoder/g0_0_10
CTOF_DEL    ---     0.260     R33C30A.D1 to     R33C30A.F1 u1_decoder/SLICE_75
ROUTE         1     0.848     R33C30A.F1 to     R33C31A.A1 u1_decoder/G_14_0_a4_0_7
CTOF_DEL    ---     0.260     R33C31A.A1 to     R33C31A.F1 u1_decoder/SLICE_72
ROUTE         1     1.458     R33C31A.F1 to     R32C30D.A0 u1_decoder/G_14_0_a4_0_13
CTOF_DEL    ---     0.260     R32C30D.A0 to     R32C30D.F0 u1_decoder/SLICE_77
ROUTE         1     1.215     R32C30D.F0 to     R32C28B.C0 u1_decoder/N_4
CTOF_DEL    ---     0.260     R32C28B.C0 to     R32C28B.F0 u1_decoder/SLICE_76
ROUTE         1     0.820     R32C28B.F0 to    R33C28B.CE u1_decoder/un1_sync_csw_2_i
(to dec_pll_clk_c)
                  --------
                    7.843   (21.5% logic, 78.5% route), 6 logic levels.
```

Report Summary This part of the report provides a matrix that illustrates the pass or fail status of the timing preferences analyzed. If all preferences are met, you will see a statement to that effect at the bottom of the summary. If not all preferences are met, each unmet preference will be marked with an asterisk in the Levels column of the summary.

```
Report Summary
--------------
------------------------------------------------------------------------------
Preference                                |   Constraint|       Actual|Levels
------------------------------------------------------------------------------
                                          |             |             |
FREQUENCY NET "up_clk_66" 66.600000 MHz   |             |             |
HOLD_MARGIN 0.100000 nS ;                 |   66.600 MHz|   74.421 MHz|   8
                                          |             |             |
FREQUENCY NET "ddr_clk" 200.000000 MHz    |             |             |
HOLD_MARGIN 0.100000 nS ;                 |  200.000 MHz|  203.046 MHz|   5
                                          |             |             |
FREQUENCY NET "clk_77" 77.760000 MHz ;    |            -|            -|   0
                                          |             |             |
FREQUENCY NET "sys_clk_125" 125.000000    |             |             |
MHz HOLD_MARGIN 0.100000 nS ;             |  125.000 MHz|  104.460 MHz|   5 *
                                          |             |             |
FREQUENCY NET "tel_clk_155" 155.520000    |             |             |
MHz HOLD_MARGIN 0.100000 nS ;             |  155.520 MHz|  139.470 MHz|   4 *
                                          |             |             |
FREQUENCY NET "tel_clk" 77.760000 MHz     |             |             |
HOLD_MARGIN 0.100000 nS ;                 |   77.760 MHz|   86.610 MHz|  12
                                          |             |             |
FREQUENCY NET "sys_clk_100" 100.000000    |             |             |
MHz HOLD_MARGIN 0.100000 nS ;             |  100.000 MHz|   98.571 MHz|  11 *
                                          |             |             |
FREQUENCY NET "ddr_rd_clk_dd"             |             |             |
200.000000 MHz HOLD_MARGIN 0.100000 nS    |             |             |
;                                         |            -|            -|   0
                                          |             |             |
FREQUENCY NET "ddr_clk270" 200.000000     |             |             |
MHz HOLD_MARGIN 0.100000 nS ;             |            -|            -|   0
                                          |             |             |
FREQUENCY NET "ddr_rd_clk_bm"             |             |             |
200.000000 MHz HOLD_MARGIN 0.100000 nS    |             |             |
;                                         |            -|            -|   0
                                          |             |             |
FREQUENCY NET "qdr_cq_clk" 250.000000     |             |             |
MHz ;                                     |            -|            -|   0
                                          |             |             |
FREQUENCY NET "qdr_clk" 250.000000 MHz    |             |             |
HOLD_MARGIN 0.100000 nS ;                 |  250.000 MHz|  207.039 MHz|   1 *

6 preferences(marked by "*" above) not met.
```

Timing Summary At the end of each report is a timing summary, which reports the following information for the design:

- the number of timing errors found
- a timing score showing total errors in picoseconds for all timing preferences
- the number of paths and connections covered by the constraints and the percentage coverage over the whole design
- a common errors matrix, which reports critical nets that are responsible for more than 10 percent of timing errors.

```
Timing summary (Hold):
---------------

Timing errors: 0  Score: 0
Cumulative negative slack: 0

Constraints cover 6 paths, 1 nets, and 12 connections (80.0%
coverage)


Timing summary (Setup and Hold):
---------------

Timing errors: 0 (setup), 0 (hold)
Score: 0 (setup), 0 (hold)
Cumulative negative slack: 0 (0+0)
```

Clock Domains Analysis The Clock Domains Analysis section helps you understand the clocking resources used by the design and the interaction between the clock domains. This section will not be included in the TRACE report if a physical preference is used.

You can use the Clock Domains Analysis section to determine the following information about your design:

- Which FREQUENCY, MULTICYCLE, MAXDELAY and BLOCK preferences are applied to a given clock domain. Note this does not include tSU, tCO, etc.
- Which clock domains transfer to another clock domain.
- The number of loads on a clock domain. A load on a clock net is simply the number of routed destination points of that clock net.

The Clock Domains Analysis section is broken down into separate subsections that report on each clock domain. A specific clock domain covered in a subsection is referred to here as a *Reported Clock Domain*. Each subsection contains information on *Transfer From Clock Domains*. A Transfer From Clock Domain starts a path that terminates in the subsection's Reported Clock Domain. So each subsection describes transfers within a specific clock domain and transfers to that specific clock domain.

Within a subsection, the term *Loads* refers to the number of synchronous elements clocked by the clock of the subsection's Reported Clock Domain. Paths to these synchronous elements may or may not originate in the Reported Clock Domain. These synchronous elements will be the end of paths that may originate from inside or outside the Reported Clock Domain.

The term *Transfers* refers to the number of synchronous elements clocked by the clock in the Transfer From Clock Domain that are the source of at least one path that terminates in the Reported Clock Domain.

Note

This Clock Domains Analysis section covers clock transfers between different physical components only. Clock transfers inside multiple clock components, such as DDR or PCS, are not covered in this section.

Below are examples from the Clock Domains Analysis section of the TRACE report and explanations:

Example 1

```
Clock Domain: txs4ls2_ck   Source: ehxplla_tx/
ehxplla_tx_0_0.CLKOS   Loads: 20
   Covered under: MULTICYCLE TO CELL "*ODDRX2A*" 2.000000 X ;
   Covered under: FREQUENCY NET "txs4ls2_ck" 250.000000 MHz ;
```

In this example, there are 20 loads on the clock domain txs4ls2_ck. The preferences that cover the data transfers of clock domain txs4ls2_ck are MULTICYCLE and FREQUENCY. There are no data transfers from another clock domain to txs4ls2_ck.

Example 2

```
Clock Domain: txs4ls4_ck   Source: TXS4LS4_CK.Q0   Loads: 26
   Covered under: FREQUENCY NET "txs4ls4_ck" 125.000000 MHz ;

   Data transfers from:
   Clock Domain: pll_sdck   Source: ehxplla_sdck/
ehxplla_sdck_0_0.CLKOS
      Blocked under: BLOCK PATH FROM CLKNET "pll_sdck" TO
CLKNET "txs4ls4_ck" ;
```

In this example, there are 26 loads on the clock domain txs4ls4_ck. The preference that covers the data transfers of clock domain txs4ls4_ck _ck is FREQUENCY. There is a data transfer from the clock domain pll_sdck to txs4ls4_ck. A BLOCK preference covers the clock domain crossing from pll_sdck to txs4ls4_ck _ck.

Example 3

```
Clock Domain: rdclki   Source: rdclk_p.PAD   Loads: 1
   No transfer within this clock domain is found
```

In this example, there is one load on the clock domain rdclki. There are no data transfers on the clock domain rdclki.

There are no Frequency, Multicycle, or Block preferences that cover the clock domain rdclki. There are no data transfers from another clock domain to rdclki.

Example 4

```
Clock Domain: txs4hs_ck   Source: ehxplla_tx/
ehxplla_tx_0_0.CLKOP   Loads: 19
   No transfer within this clock domain is found

   Data transfers from:
   Clock Domain: txs4ls2_ck   Source: ehxplla_tx/
ehxplla_tx_0_0.CLKOS
     Covered under: FREQUENCY NET "txs4hs_ck" 500.000000 MHz ;
Transfers: 1
```

In this example, there are 19 loads on the clock domain txs4ls4_ck. There are no data transfers on the clock domain txs4hs_ck. There are no FREQUENCY, MULTICYCLE, or BLOCK preferences that cover the clock domain txs4hs_ck. There is a data transfer from the clock domain txs4ls4_ck to txs4hs_ck. A FREQUENCY preference covers the clock domain crossing from txs4ls4_ck to txs4hs_ck and there is one data transfer.

Example 5

```
Clock Domain: pll_sdck   Source: ehxplla_sdck/
ehxplla_sdck_0_0.CLKOS   Loads: 413
   Covered under: FREQUENCY NET "pll_sdck" 150.000000 MHz ;

   Data transfers from:
   Clock Domain: txs4ls2_ck   Source: ehxplla_tx/
ehxplla_tx_0_0.CLKOS
      Not reported because source and destination domains are
unrelated.
      To report these transfers please refer to preference
CLKSKEWDIFF to define
      external clock skew between clock ports.
```

In this example, the clock domain pll_sdck and txs4ls2_ck are driven by two unrelated external clocks. Because there are no skew relationships defined between the external clocks, these two clock domains do not have a meaningful skew. Although there are data transfers between these two domains, they will not be reported by default. Preference CLKSKEWDIFF can be used on the external clocks in order to report these transfers.

Interclock Domain Transfers Between Unrelated Clocks
Two clock signals are defined as unrelated if they are not driven by the same clock input pad and you have not defined a relationship between them using the CLKSKEWDIFF preference.

Prior to ispLEVER 7.2, data transfers between unrelated clocks were reported as if those clocks were related and had a skew of 0. In ispLEVER 7.2 and later, data transfers between unrelated clocks are not reported. If you want to have those transfers reported, they must define a clock relationship between the two external clocks using the CLKSKEWDIFF preference. This updated behavior in TRACE avoids needlessly over-constraining a design. For example, data crossing between two asynchronous clocks should be analyzed during simulation. Static timing analysis cannot be used to guarantee proper operation. The former default constraint adds no value, and could negatively impact design performance.

The following is an example from the Clock Domains Analysis section, where two clocks are considered unrelated and the data crossing between them is not reported.

```
Clock Domain: Comclk_i_int   Source: Comclk_i.PAD
    Not reported because source and destination domains are
    unrelated.
    To report these transfers please refer to preference
    CLKSKEWDIFF to define external clock skew between clock
    ports.
```

The Clock Domains Analysis section will list all data transfers between clocks that are not reported as the clocks are considered unrelated. You should verify that all clock pairs considered unrelated are actually unrelated. A good design practice is to use simulation to verify the proper operation of data

crossing unrelated clock domains. The following is an example of a Clock Domains Analysis section.

```
Clock Domains Analysis
------------------------
Found 37 clocks:
Clock Domain: pll_sdck   Source: ehxplla_sdck/ehxplla_sdck_0_0.CLKOS   Loads: 696
   Covered under: FREQUENCY NET "pll_sdck" 150.000000 MHz ;

   Data transfers from:
   Clock Domain: txs4ls4_ck   Source: ehxplla_tx/ehxplla_tx_0_0.CLKOS
      Not reported because source and destination domains are unrelated.
      To report these transfers please refer to preference CLKSKEWDIFF to define
      external clock skew between clock ports.

   Clock Domain: rdclki   Source: rdclk_p.PAD
      Not reported because source and destination domains are unrelated.
      To report these transfers please refer to preference CLKSKEWDIFF to define
      external clock skew between clock ports.

   Clock Domain: iobuf/iobuf_tstatus_ckii   Source: tstatus_ck.PAD
      Not reported because source and destination domains are unrelated.
      To report these transfers please refer to preference CLKSKEWDIFF to define
      external clock skew between clock ports.

   Clock Domain: pll_sdck2   Source: ehxplla_sdck/ehxplla_sdck_0_0.CLKOP
      Covered under: FREQUENCY NET "pll_sdck" 150.000000 MHz ;   Transfers: 1

   Clock Domain: spi4_256ch/rxs4ls2_ck   Source: spi4_256ch/gearbox/rxgb/
CLKDIV_RXIO.CLKO
      Not reported because source and destination domains are unrelated.
      To report these transfers please refer to preference CLKSKEWDIFF to define
      external clock skew between clock ports.

   Clock Domain: txs4ls2_ck   Source: spi4_256ch/gearbox/txgb/CLKDIV_TXIO.CLKO
      Not reported because source and destination domains are unrelated.
      To report these transfers please refer to preference CLKSKEWDIFF to define
      external clock skew between clock ports.

Clock Domain: rxlt10_trerr   Source: spi4_256ch/s4dp/deskew/SLICE_1395.Q0   Loads: 2
   No transfer within this clock domain is found

   Data transfers from:
   Clock Domain: txs4ls4_ck   Source: ehxplla_tx/ehxplla_tx_0_0.CLKOS
      Covered under: FREQUENCY PORT "txref" 100.000000 MHz ;   Transfers: 1

Clock Domain: rxlt40_trerr   Source: spi4_256ch/s4dp/deskew/SLICE_1396.Q0   Loads: 2
   No transfer within this clock domain is found

   Data transfers from:
   Clock Domain: txs4ls4_ck   Source: ehxplla_tx/ehxplla_tx_0_0.CLKOS
      Covered under: FREQUENCY PORT "txref" 100.000000 MHz ;   Transfers: 1

Clock Domain: N_105_iZ0   Source: SLICE_2784.F0   Loads: 1
   No transfer within this clock domain is found

   Data transfers from:
   Clock Domain: txs4ls4_ck   Source: ehxplla_tx/ehxplla_tx_0_0.CLKOS
      Not reported because source and destination domains are unrelated.
      To report these transfers please refer to preference CLKSKEWDIFF to define
      external clock skew between clock ports.
```

Loop Report This section of the TRACE report lists all paths that cannot be analyzed because of asynchronous loops in the circuit. This loop report is available in both Error and Verbose levels of reporting. Path name, fanout, delay (ns), destination, and source information are provided.

See Also ▶"Viewing TRACE Reports" on page 700

- "Strategies for Timing Analysis" on page 695
- "Setting Options in Timing Analysis View" on page 733

Cross-Probing to Synplify Pro HDL Analyst

Lattice Diamond, together with the Synplify Pro for Lattice software, enables you to cross-probe a path from the TRACE report to HDL Analyst view. This provides you with a schematic view of the path and helps you diagnose problem areas to improve the timing.

To cross-probe from TRACE to the Synplify Pro HDL Analyst:

1. In Diamond, run the **Place & Route Trace** process.
2. Click the Synplify Pro button on the toolbar to launch Synplify Pro for Lattice.
3. In the Synplify Pro main window, click the **Run** button to synthesize the project and create the schematic views for the active implementation.
4. Click the Technology View button on the Synplify Pro toolbar to open HDL Analyst.
5. Choose **File > Open**, navigate to the TRACE report file (.twr) that you generated in Step 1, and open it.

   The Synplify Pro Text Editor View opens and displays the TRACE report.
6. In the Text Editor View, highlight the path in the TRACE report that you want to examine by dragging your mouse over it.
7. Right-click the selected path and choose **Filter in Analyst** from the pop-up menu.
8. Click the Technology View tab at the bottom to return to the HDL Analyst view.

   A schematic view of the selected path is displayed in HDL Analyst.

To view the critical path:

- Right-click anywhere in HDL Analyst and choose **Show Critical Path**.

See Also ▶"Preference Details" on page 705

TRACE Timing Arc Descriptions

The following table provides definitions of timing arcs for setup time, hold time, delays, and minimum pulse width that you encounter within the TRACE report (.tw1/.twr) file. Timing arcs refer to intra-cell path delays—the time it takes for a signal to flow from a given logic cell's input transition to a resulting output rise (or fall) output transition. TRACE timing analysis reports timing arcs as a part of constraint calculation.

Table 2: Setup Time Timing Arcs in TRACE

Identifier	Description
ADDR_SET	Address setup
CE_SET	CE setup
CLK2ECLK_SET	ODDR CLK to Edge CLK setup
CNTL_SET	DQSDLL UDDCNTL pin to CLK setup
CS_SET	EBR CS to CLK setup
DATA_SET	EBR DI (data in) setup
DIN_SET	SLICE DIN to CLK setup
DI_SET	IOLOGIC DI to CLK setup
ECLK2CLK_SET	IDDR Edge CLK to CLK setup
LSRREC_SET	LSR Recovery setup
LSR_SET	LSR setup
M_SET	SLICE/FSLICE M pins to CLK setup
ONEG0_SET	IOLOGIC ONEG0 to CLK setup
ONEG1_SET	IOLOGIC ONEG1 to CLK setup
ONEG2_SET	IOLOGIC ONEG2 to CLK setup
OPOS0_SET	IOLOGIC OPOS0 to CLK setup
OPOS1_SET	IOLOGIC OPOS1 to CLK setup
OPOS2_SET	IOLOGIC OPOS2 to CLK setup
RST_SET	Reset to CLK setup
SMI_SET	DLL SMI address pins to SMICLK setup
TCK_SET	JTAG JTDO pin to TCK setup.
TD_SET	IOLOGIC TD pin to CLK setup
WAD_SET	SLICE WAD pin to WCK setup
WD_SET	SLICE WD pin to WCK setup
WE_SET	EBR WE to CLK setup
WRE_SET	SLICE WRE pin to WCK setup

Table 3: Hold Time Timing Arcs in TRACE

Identifier	Description
ADDR_HLD	Address hold time
CE_HLD	Clock Enable (CE) hold
CLK2ECLK_HLD	IOLOGIC CLK to edge CLK hold
CNTL_HLD	DQSDLL UDDCNTL pin to CLK hold
CS_HLD	EBR CS to CLK hold
DATA_HLD	EBR data hold
DIN_HLD	SLICE DIN to CLK hold
DI_HLD	IOLOGIC DI to CLK hold
ECLK2CLK_HLD	IDDR edge CLK to CLK hold
LSR_HLD	LSR hold
M_HLD	SLICE/FSLICE M pins to CLK hold
ONEG0_HLD	IOLOGIC ONEG0 to CLK hold
ONEG1_HLD	IOLOGIC ONEG1 to CLK hold
ONEG2_HLD	IOLOGIC ONEG2 to CLK hold
OPOS0_HLD	IOLOGIC OPOS0 to CLK hold
OPOS1_HLD	IOLOGIC OPOS1 to CLK hold
OPOS2_HLD	IOLOGIC OPOS2 to CLK hold
RST_HLD	Reset to CLK hold
SMI_HLD	DLL SMI address pins to SMICLK hold
TCK_HLD	JTAG JTDO pin to TCK hold
TD_HLD	IOLOGIC TD pin to CLK hold
WAD_HLD	SLICE WAD pin to WCK hold
WD_HLD	SLICE WD pin to WCK hold.
WE_HLD	EBR WE to CLK hold
WRE_HLD	SLICE WRE pin to WCK hold

Table 4: Delay Timing Arcs in TRACE

Identifier	Description
C0TOFCO_DEL	SLICE/FSLICE A0/B0/C0/D0 to FCO delay
C1TOFCO_DEL	SLICE/FSLICE A1/B1/C1/D1 to FCO delay
C2OUT_DEL	IOLOGIC CLK to INFF delay
C2Q_DEL	DSQCLK to Q delay
CK2DVAL_DEL	DQS CLK to DATAVALID delay
CK2OUT_DEL	DLL CLKI or CLKFB to CLKOP/CLKOS; CLKI to CLKO; SMICLK to SMIRDATA.
CLKI2OP_DEL	PLL CLKI to CLKOP delay
CLKI2OS_DEL	PLL CLKI to CLKOS delay
CLKI2OK_DEL	PLL CLKI to CLKOK delay
CLKFB2IFB_DEL	PLL CLKFB to CLKINTFB delay
CLKI2IFB_DEL	PLL CLKI to CLKFB delay
CLKFB2OK_DEL	PLL CLKFB to CLKOK delay
CLKFB2OP_DEL	PLL CLKFB to CLKOP delay
CLKFB2OS_DEL	PLL CLKFB to CLKOS delay
CLKOUT_DEL	CLKDIV CLKI to CDIV* or RELEASE to CDIV* delay
CLKTOF_DEL	SLICE WCK to F0 or F1 delay
CTOF1_DEL	SLICE/FSLICE A0,B0,C0,or D0 to F1 delay
CTOF_DEL	SLICE/FSLICE A0,B0,C0,D0 to F0 or A1,B1,C1,D1 to F1 delay.
CTOOFX_DEL	SLICE/FSLICE A0,B0,C0,D0 to OFX0 or A1,B1,C1,D1 to OFX0 delay.
DOPAD_DEL	PIO IOLDO to PAD delay
DQI2DQC_DEL	DQS DQSI to DQSC delay
DQI2DQO_DEL	DQS DQSI to DQSO delay
DQI2PT_DEL	DQS DQSI to PRMBDET delay
DQSI2POL_DEL	DQS DQSI to DDRCLKPOL delay
DTPAD_DEL	PIO IOLTO to PAD delay
FCITOF0_DEL	SLICE/FSLICE FCI to F0 delay
FCITOF1_DEL	SLICE/FSLICE FCI to F1 delay
FCITOFCO_DEL	SLICE/FSLICE FCI to FCO delay
FXTOOFX_DEL	SLICE/FSLICE FXA or FXB to OFX1 delay

Table 4: Delay Timing Arcs in TRACE (Continued)

Identifier	Description
JTCK_DEL	JTAG TCK to JTCK delay
LSR2Q_DEL	SLICE/FSLICE LSR to Q0 or Q1 delay
MTOOFX_DEL	SLICE/FSLICE M0 to OFX0 or M1 to OFX1 delay
MUX_DEL	DCS CLK0 or CLK1 to DCSOUT delay
PADI_DEL	PIO PAD to PADDI delay
REG_DEL	SLICE/FSLICE CLK to Q0 or Q1 delay
RST2LOCK_DEL	DQSDLL RST to LOCK delay
RST_DEL	DLL RSTN to LOCK delay
TLATCH_DEL	IOLOGIC DI to INFF or ONEG0 to IOLDO or TD to IOLTO delay
ZERO_DEL	Connection with zero delay
LSR_OUT	IOLOGIC LSR to INFF, IOLDO, or IOLTO delay

Table 5: MPW (Minimum Pulse Width) Timing Arcs in TRACE

Identifier	Description
CLKH_MPW	DLL CLKI, CLKFB, or SMICLK minimum high pulse width
CLKL_MPW	DLL CLKI, CLKFB, or SMICLK minimum low pulse width
CLK_MPW	CLK minimum pulse width
LSR_MPW	DLL RSTN or SMIRSTN minimum pulse width
PAD_MPW	PIO PAD minimum pulse width
RST_MPW	EBR RSTA or RSTB minimum pulse width

Running I/O Timing Analysis

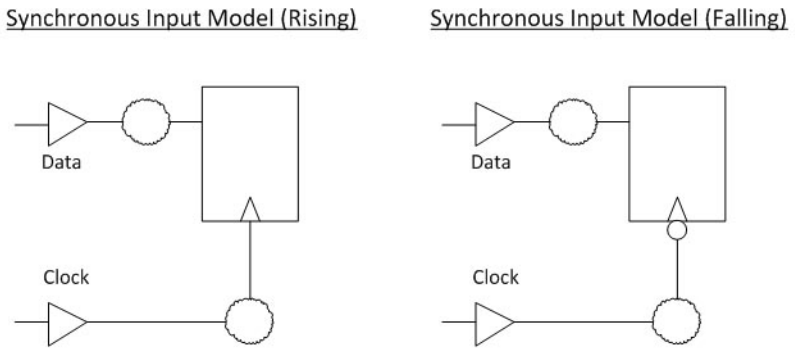
The I/O Timing Analysis process generates a report (.ior) of worst case input setup/hold and min/max clock-to-out I/O results for the design. For each input data port in the design, it provides the setup and hold time requirements in relation to an input clock port. For each output data port or forwarded clock in the design, it provides the minimum and maximum clock-to-out timing in relation to an input clock port.

Unlike the TRACE report, the I/O Timing Analysis report is not limited to reporting paths covered by timing preferences. The report is focused on FPGA I/O timing characteristics for the full range of the device performance grades. The final numbers reported are based on the worst-case timing across all of the performance grades. To find the details on any of the ports listed, the TRACE report will need to be utilized.

What Gets Analyzed in I/O Timing Analysis

I/O Timing Analysis reports on FPGA I/O port timing. The following are descriptions of the I/O models, showing how they are presented in the report file (.ior).

Synchronous Input Model The following diagrams show rising edge and falling edge synchronous input timing models. They will be analyzed by I/O Timing Analysis, which will provide the worst-case setup and hold requirements.



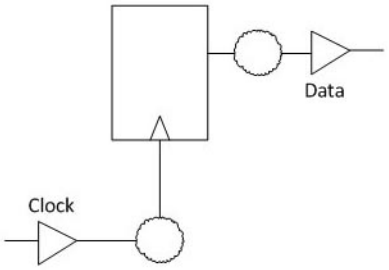
The rising edge and falling edge timing models are shown differently in the .ior report file, as shown in the following example.

```
Port     Clock     Edge  Setup Performance_Grade  Hold Performance_Grade
------------------------------------------------------------------------
din_f    clkin     F    -0.855      M            2.344     7
din_r    clkin     R    -0.855      M            2.344     7
```

In the example, port “din_f” is clocked on the falling edge of clock input port “clkin”. The worst-case setup time for port “din_f” with respect to clock input “clkin” is -0.855ns. A negative input setup time means that the data will not be latched by the clock until after the clock edge. This type of timing happens often in today’s FPGAs with flip-flops located in or near the I/O. For this worst-case input setup, the device performance grade was the –M. The –M performance grade is the fastest performance grade possible for this device. All other performance grades have an input setup which is less than -0.855ns.

On the hold timing side, input port “din_f” must hold its data until 2.344ns after the input clock “clkin”. This worst-case hold timing was found in the -7 performance grade. All other performance grades hold timing is less than the 2.344ns requirement.

Synchronous Output Model The following diagram shows a synchronous output timing model. This model will be analyzing by I/O Timing Analysis and provide the worst-case clock-to-out requirements.



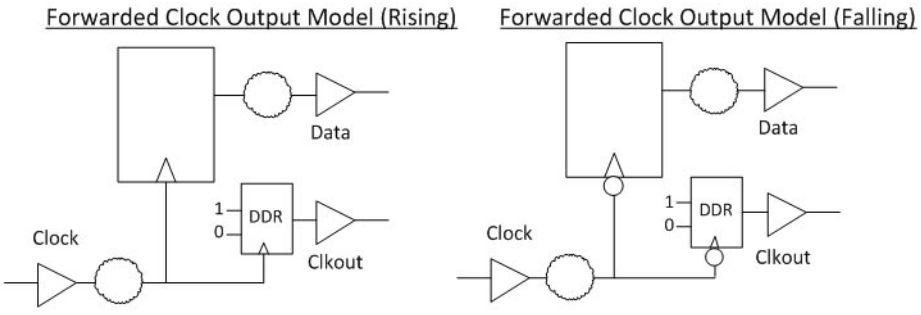
The synchronous output model produces a minimum and maximum clock-to-out line in the .ior report, as shown in the following example:

```
Port       Clock     Edge  Max_Delay Performance_Grade  Min_Delay Performance_Grade
------------------------------------------------------------------------------
dout       clkin      R     7.031         7               3.365          M
```

In this example, port “dout” is clocked on the rising edge of clock input port “clkin”. The worst-case maximum delay from the clock input to the data out time is 7.031ns using a -7 performance grade. All other performance grades have a faster clock-to-out time.

On the minimum side, using –M performance grade, the clock-to-out is reported as 3.365ns. All other performance grades have a longer clock-to-out time.

Forwarded Clock Output Model The following diagram shows rising edge and falling edge forwarded clock output models. Forwarded clocks are commonly used in today’s FPGA I/O parallel interfaces.



Both of these models use a DDR element as the source of the forwarded output clock. This is the recommended method for using a forwarded clock,

because it keeps the clock routing consistent for the data as the clock output. This way, both the data and the clock go through the same logic in the I/O, which results in minimal skew between the clock output and data.

Note

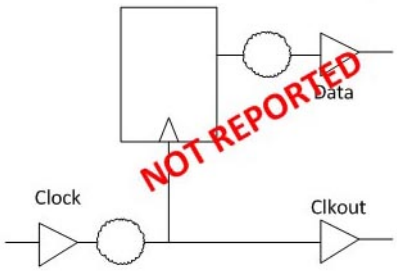
To find out about specific Lattice FPGA architectures and high-speed interface recommendations, visit the Lattice website at www.latticesemi.com and search for device-specific application notes.

To understand the relationship between the data output and forwarded clock, it is important to examine both in the .ior report, as shown in the following example. Each output port will show a clock-to-out number with respect to the input clock. The difference in the clock-to-out will be the skew between the data and clock.

```
Port        Clock     Edge  Max_Delay Performance_Grade  Min_Delay Performance_Grade
------------------------------------------------------------------------------------
clkout      clkin       R     7.031         7               3.365          M
dout        clkin       R     7.031         7               3.365          M
```

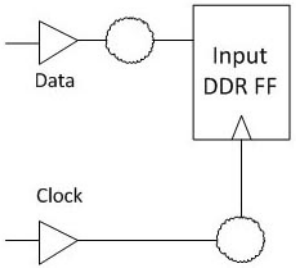
In this example, the clock output “clkout” takes 7.031ns from the input clock “clkin,” and the data output “dout” takes exactly the same time, resulting in an aligned relationship.

It is important to note that the forwarded clock analysis can only happen using the model shown in “Forwarded Clock Output Model” on page 724. The following traditional forwarded clock timing model will not be reported properly in the I/O Timing Analysis .ior report file, because there is no clocked element on the forwarded clock output.



This non-supported timing model will always result in a relatively large amount of skew between the output data and the forwarded clock, since different logical paths are used. The data path will add a clock-to-out delay of the output flip-flop. The clock output path will add a routing delay to an output pin. These paths can have a relatively large difference in timing.

Double-Data-Rate Input Model The following diagram shows a double-data-rate (DDR) input timing model. This model will be analyzed by the I/O Timing Analysis and provide the worst-case setup and hold requirements.



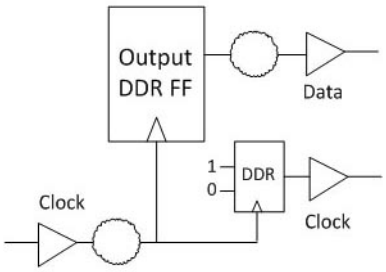
Input DDR interfaces latch data on both the rising and falling edge of the clock. Traditionally, there are two modes of an input DDR interface: centered and aligned. A centered interface has the clock edge centered in the middle of the data valid of the data input before entering the device. An aligned interface has the clock edge and data edge aligned together. Depending on the interface mode, setup and hold numbers need to be understood differently to apply them to the system design. Each current Lattice FPGA architecture family has an application note on designing high-speed interfaces that explains how to use the setup and hold numbers reported in the I/O analysis report.

The following is an example report line for an input DDR interface.

Port	Clock	Edge	Max_Delay	Performance_Grade	Min_Delay	Performance_Grade
din_ddr	clkin_ddr	R	0.234	8	0.258	7

Double-Data-Rate Output Model The following diagram shows a double-data-rate (DDR) output timing model. Like the forwarded clock model, a DDR element is used to produce the clock output. This ensures that the same logical path is used for both the data and the clock output. This model will be

analyzed by I/O Timing Analysis and provide the worst-case maximum and minimum delay requirements.



Output DDR interfaces launch data on both the rising and falling edge of the clock. Traditionally, there are two modes of an output DDR interface: centered and aligned. A centered interface has the clock edge centered in the middle of the data valid of the data output when leaving the device. An aligned interface has the clock edge and data edge aligned together. Using the I/O analysis report, you can identify the relationship of the clock output and the data output to determine the skew of an aligned or centered interface.

Below is an example report line for an output DDR interface.

Port	Clock	Edge	Max_Delay	Performance_Grade	Min_Delay	Performance_Grade
clkout	clkin_ddr	R	7.031	7	3.365	M
dout_ddr	clkin_ddr	R	7.031	7	3.365	M

Generating the I/O Timing Analysis Report

There are two options for generating the I/O Timing Report: you can generate a report with only the worst-case I/O timing reported across all performance grades, or you can generate a report that includes the worst-case I/O timing at the top followed by the results of each available performance grade of the device.

All Performance Grade = False By default, the IO Timing Analysis "All Performance Grade" option in the active strategy is set to False. When the setting of False is used, the given performance grade and any grade faster than the given performance grade are analyzed for each port of the design, and the worst case across all the analyzed grades is reported. These analyzed performance grades include the minimum (M) grade, which is the virtual performance grade that is the fastest with minimum delay. The generated report includes the summary only.

To generate the default I/O Timing Analysis report:

- In the Process View, double-click **I/O Timing Analysis**.

  Diamond computes the data of the given performance grade and any faster performance grades for the device and generates a summary report.

All Performance Grade = True When the IO Timing Analysis "All Performance Grade" option in the active strategy is set to True, all performance grades (including M) of the target device are analyzed, and the worst case across all the grades is reported. The generated report includes the summary, followed by a detailed report for each grade.

To generate the detailed I/O Timing Analysis report:

1. Choose **Project > Active Strategy > IO Timing Analysis Settings**.

   The Strategy dialog box opens and displays the IO Timing Analysis settings.

2. In the IO Timing Analysis pane, change the All Performance Grade value from False to **True** and click **OK**.

3. In Diamond, select the Process tab and double-click **I/O Timing Analysis**.

   Diamond computes the data over all available performance grades and generates a detailed report and summary.

Viewing the I/O Timing Analysis Report The I/O Timing Analysis Report becomes available in the Reports Window as soon as the I/O Timing Analysis process has finished.

To view the I/O Timing Analysis report:

- In the Reports view, expand the Analysis Reports folder and select **I/O Timing Analysis**.

See Also ▶"Running I/O Timing from the Command Line" on page 2531

Example I/O Timing Analysis Reports

The following I/O Timing Report was generated using the default setting of False for "All Performance Grade" in the active strategy. The report includes analysis of the given performance grade and any grade faster than the given performance grade.

Figure 6: I/O Timing Analysis with All Performance Grade = False

```
I/O Timing Report
// Design: top
// Package: FPBGA484
// ncd File: top20_top20.ncd
// Version: Diamond Version 2.0.0.99
// Written on Tue Apr 24 10:03:35 2012
// M: Minimum Performance Grade
// iotiming top20_top20.ncd top20_top20.prf

I/O Timing Report (All units are in ns)

Worst Case Results across Performance Grades (M, 9, 8, 7):

// Input Setup and Hold Times

Port     Clock     Edge  Setup Performance_Grade  Hold Performance_Grade
------------------------------------------------------------------------
din_ddr  clkin_ddr R     0.234      8             0.258     7
din_f    clkin     F    -0.855      M             2.344     7
din_r    clkin     R    -0.855      M             2.344     7


// Clock to Output Delay

Port       Clock     Edge  Max_Delay Performance_Grade  Min_Delay Performance_Grade
------------------------------------------------------------------------------------
clkout_ddr clkin_ddr R     7.031         7             3.365          M
clkout_f   clkin     F     7.031         7             3.365          M
clkout_r   clkin     R     7.031         7             3.365          M
dout_ddr   clkin_ddr R     7.031         7             3.365          M
dout_f     clkin     F     6.849         7             3.264          M
dout_r     clkin     R     6.849         7             3.264          M
```

The following I/O Timing Report was generated using the setting of True for "All Performance Grade" in the active strategy. The report includes analysis of all performance grades.

Figure 7: I/O Timing Analysis with All Performance Grade = True

```
// Design: top
// Package: FPBGA484
// ncd File: top20_top20.ncd
// Version: Diamond Version 2.0.0.99
// Written on Mon Apr 23 15:28:50 2012
// M: Minimum Performance Grade
// iotiming -v top20_top20.ncd top20_top20.prf

I/O Timing Report (All units are in ns)

Worst Case Results across All Performance Grades (M, 9, 8, 7):

// Input Setup and Hold Times

Port  Clock Edge  Setup Performance_Grade  Hold Performance_Grade
------------------------------------------------------------------
din_f clkin F    -0.855      M       2.344     7
din_r clkin R    -0.855      M       2.344     7


// Clock to Output Delay

Port     Clock Edge  Max_Delay Performance_Grade  Min_Delay Performance_Grade
------------------------------------------------------------------------
clkout_f clkin F     7.031         7        3.365          M
clkout_r clkin R     7.031         7        3.365          M
dout_f   clkin F     6.849         7        3.264          M
dout_r   clkin R     6.849         7        3.264          M

========================================================================

Detailed Analysis of Each Performance Grade:


// Input Setup/Hold Times (Performance Grade: M)

Port  Clock Edge    Setup     Hold
------------------------------------------------------------
din_f clkin F      -0.855     1.017
din_r clkin R      -0.855     1.017


// Clock to Output Delay (Performance Grade: M)

Port     Clock Edge    Max_Delay   Min_Delay
-----------------------------------------------------------
clkout_f clkin F       4.055       3.365
clkout_r clkin R       4.055       3.365
dout_f   clkin F       3.954       3.264
dout_r   clkin R       3.954       3.264
```

Figure 7: I/O Timing Analysis with All Performance Grade = True (Continued)

```
// Input Setup/Hold Times (Performance Grade: 9)

Port  Clock Edge    Setup     Hold
------------------------------------------------------------
din_f clkin F      -1.618     1.975
din_r clkin R      -1.618     1.975


// Clock to Output Delay (Performance Grade: 9)

Port     Clock Edge    Max_Delay   Min_Delay
------------------------------------------------------------
clkout_f clkin F       6.111       5.829
clkout_r clkin R       6.111       5.829
dout_f   clkin F       5.949       5.667
dout_r   clkin R       5.949       5.667


// Input Setup/Hold Times (Performance Grade: 8)

Port  Clock Edge    Setup     Hold
------------------------------------------------------------
din_f clkin F      -1.747     2.160
din_r clkin R      -1.747     2.160


// Clock to Output Delay (Performance Grade: 8)

Port     Clock Edge    Max_Delay   Min_Delay
------------------------------------------------------------
clkout_f clkin F       6.570       6.254
clkout_r clkin R       6.570       6.254
dout_f   clkin F       6.398       6.082
dout_r   clkin R       6.398       6.082


// Input Setup/Hold Times (Performance Grade: 7)

Port  Clock Edge    Setup     Hold
------------------------------------------------------------
din_f clkin F      -1.876     2.344
din_r clkin R      -1.876     2.344


// Clock to Output Delay (Performance Grade: 7)

Port     Clock Edge    Max_Delay   Min_Delay
------------------------------------------------------------
clkout_f clkin F       7.031       6.677
clkout_r clkin R       7.031       6.677
dout_f   clkin F       6.849       6.495
dout_r   clkin R       6.849       6.495
```

Using Timing Analysis View

The Timing Analysis view is a graphical user interface for the static timing analysis program, TRACE. The Timing Analysis view's main window allows you to view the path delay tables and TRACE report of your timing preferences after placement and routing. Additionally, Timing Analysis View includes a TPF version of Spreadsheet View, which allows you to experiment with different sets of timing preferences through the use of timing preference files (.tpf).

Timing Analysis View Main Window Timing Analysis View's main window provides multiple panes, path delay tables, and cross-probing features for viewing key timing data in your design, as well as the start points, end points, and categories of unconstrained paths. For example, you can click on a preference you defined on a particular clock net or port and see the slack associated with it, the delay value, and the source and destination pins. Or you can click on a listed preference for an unconstrained path and view the start point, end point, classification, and clocks. In addition, you can cross-probe a timing path to Floorplan View or Physical View. This can be useful for reconfiguring or regrouping elements to optimize the timing on a critical path.

TPF Spreadsheet View The TPF version of Spreadsheet View provides a way for you to modify timing preferences and see the results without having to rerun Place & Route. After making changes, you simply click the Update button in the Timing Analysis View main window to examine the updated report and delay tables. When you save the changes, they are written to a timing preference file (.tpf) so that your project's logical preference source file (.lpf) is not affected. You can experiment with multiple sets of .tpf files, compare the results, and associate .tpf files with different implementations. You can later copy the modified preferences to your project's .lpf file.

See Also
- "Modifying Preferences for Timing Analysis" on page 734
- "Cross-Probing from Timing Analysis View" on page 753
- "Rearranging the Timing Analysis View Layout" on page 745

Opening Timing Analysis View

Timing Analysis View becomes available after placement and routing. When you first open Timing Analysis View, it displays timing information based on constraints in the HDL source files and the active logical preference file (.lpf). After you have created a timing preference file (.tpf) and set it as active, Timing Analysis View will display the .tpf file information when it opens.

To open Timing Analysis view:

- In Diamond, choose **Tools > Timing Analysis View**.

  A progress indicator message box opens, showing that Diamond is calculating the delays. When the progress indicator closes, the timing

preferences, settings, and analysis are displayed in the delay path tables, schematic, and report panes to the right.

Note

Each time you start Timing Analysis view, Diamond loads the NCD file with default timing constraints from the active timing preference file (.tpf), and it does all the necessary calculation. The first time you start Timing Analysis View, it appears to take longer.

To prevent a performance lag, 30 worst case paths are automatically cached in memory for each timing preference even if you have this option set to a lower number. This behavior is set up so that the tool does not have to spend performance time recalculating timing. If you want to analyze more than 30 worst case paths, or if you change any other parameters such as performance grade for setup analysis, the Timing Analysis view will recalculate all paths.

See Also

- "Modifying Preferences for Timing Analysis" on page 734
- "Importing In-Memory Timing Preferences" on page 736
- "Setting Options in Timing Analysis View" on page 733
- "Rearranging the Timing Analysis View Layout" on page 745

Setting Options in Timing Analysis View

The Settings pane of Timing Analysis View displays the timing analysis settings in blue font. You can change the values of these settings for your current session of Timing Analysis view by opening the Settings dialog box.

To change the timing analysis settings:

1. Click the **Settings**  **button** on the vertical toolbar.

   Optionally, right-click anywhere inside the Settings pane, and choose **Settings** from the pop-up menu.
2. In the dialog box, select a performance grade for Setup and Hold analysis.
3. Select one or more of the following options:
   - Check Unconstrained Connections – lists the paths that are not covered by any timing preference. This legacy option will be discontinued after the next two Diamond releases. It is recommended that you use the Check Unconstrained Paths option instead.
   - Check Unconstrained Paths – reports paths that are not constrained and shows the start point and end point of each path. In the report, TRACE will suggest some timing preferences to constrain the given paths. If you select this option, you will be able to view the unconstrained paths in Timing Analysis View by category and cross-probe to Floorplan View, Physical View, and the TRACE report.

     Based on the design and the required performance, only necessary paths should be constrained so that PAR focuses only on the optimization of the important paths. However, the Unconstrained Paths section of the TRACE report is very useful for identifying

missing timing constraints that could be important to the design. This option does not require you to add more preferences in an attempt to constrain all paths. Instead, it serves as a reminder that there might be a necessary preference that is missing, which could impact the desired performance of the design. When you select this option, “Unconstrained Paths” will be added to the “Other Reports” section in the Preferences pane of Timing Analysis View’s main window. See also Important points about connection coverage.

Note

The Check Unconstrained Paths option cannot be used with the -allprefpath command-line option.

- Report Asynchronous Timing Loops – reports paths that cannot be analyzed because they are asynchronous loops.
- Full Name – reports the full-length component names instead of truncated names.

4. Select the Report Style.

   The Verbose Timing Report, which is the default, lists detailed logic and route delays for all constrained paths and nets.

   The Error Timing Report lists logic and route delays for each path and net that causes a timing error.

5. Select the number of worst-case paths to be reported for each timing preference. You can select a value from 1 to 200. The default is 10.
6. If you selected the Check Unconstrained Paths option, select the number of unconstrained paths to be reported.
7. Click **OK**.

   The Settings pane in Timing Analysis View displays the settings you selected.

See Also ▶“Setting TRACE Options” on page 696

Modifying Preferences for Timing Analysis

The TPF version of Spreadsheet View enables you to make modifications to your timing preferences for the purpose of experimentation with timing analysis. After you modify timing preferences in the TPF Spreadsheet View, you can view the results by updating them in the Timing Analysis View main window. You can then save the modified preferences to a timing preference file (.tpf).

To modify preferences for timing analysis:

1. From the Timing Analysis View main window, choose **Edit > TPF Preferences** or click the Spreadsheet View button on the vertical toolbar.

2. In the TPF Spreadsheet View, select the tab for the preference you want to define:
   - On the Global Preference sheet, double-click the cell for the preference value you want to change and enter the new value.
   - On the Timing Preference sheet, double-click the desired preference type and select the desired options in the dialog box.
   - On the Group sheet, double-click the desired Define Group type to open the dialog box. Specify the name and select the desired options.
3. Return to the Timing Analysis View main window.

   The tabs of Timing Analysis View and Spreadsheet View – TPF are now marked with asterisks, indicating that changes have been made that have not yet been saved to a .tpf file.

   The Update button  on the toolbar is rotating, which indicates that timing preference changes need to be updated for timing analysis. Any time a data source changes after the view is initialized, the view will detect this condition and prompt you to refresh the changes with the Update button.

   Note

   Updating the timing preference changes does not affect your project's logical preference file.

4. Click the **Update** button.

   The button immediately becomes dim and a progress indicator message appears, showing that Diamond is calculating the results.

   When the progress indicator closes, you can view the results in the delay path tables, schematic, and report of the Timing Analysis View main window.

Note

The asterisks will remain on the tabs of Timing Analysis View and the TPF Spreadsheet View until you save the modified timing preferences to a .tpf file.

When you update your preference changes, they are updated in system memory in your Diamond session so that the views all display the same information. These changes are not automatically written to your current .tpf file. You must save them to a .tpf file separately. You can do this before or after updating the view. See "Creating a New Timing Preference File" on page 747.

See Also "Importing In-Memory Timing Preferences" on page 736

"Creating a New Timing Preference File" on page 747

Importing In-Memory Timing Preferences

When you first open Timing Analysis View, it displays the timing preferences that are in the logical preference file (.lpf). It also displays any unsaved timing preferences that you created in the regular Spreadsheet View after running Place & Route. These in-memory timing preferences get added automatically to the timing preference file (.tpf) when you save them in Timing Analysis View.

However, if you edit timing preferences in the regular Spreadsheet View after you have launched Timing Analysis View, you must import these in-memory timing preferences in order to run timing analysis on them.

To import in-memory timing preferences:

1. In the Timing Analysis View main window, choose **Edit > TPF Preferences** or click the Spreadsheet View button on the vertical toolbar.
2. In the TPF Spreadsheet View, choose **File > Import > Copy LPF to TPF**.
3. Return to the Timing Analysis View main window.

   The Update button on the toolbar is rotating, which indicates timing changes that need to be updated for timing analysis.
4. Click the **Update** button.

   When the progress indicator closes, you can view the results in the delay path tables, schematic, and report of the Timing Analysis View main window.

See Also ▶"Modifying Preferences for Timing Analysis" on page 734

Timing Analysis View Features

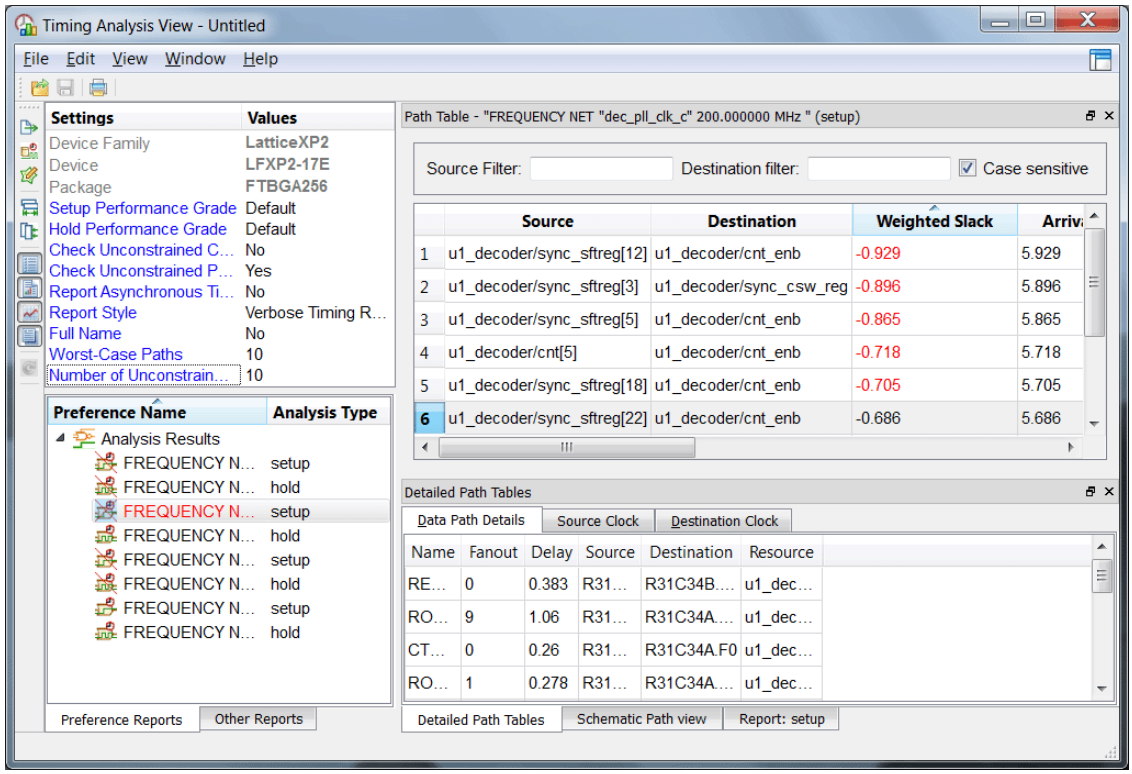
Timing Analysis View provides two main views: a main window consisting of multiple panes for viewing timing preferences, path tables, and the TRACE report; and a TPF Spreadsheet View consisting of Global, Timing, and Group preference sheets. The main window enables you to examine the results of timing preferences, cross-probe delay paths to Floorplan View and Physical View, and troubleshoot paths of the design that restrict performance. The TPF Spreadsheet View enables you to modify timing preferences and then return to the Timing Analysis View main window to examine the results and save the changes to one or more timing preference files (.tpf).

See Also ▶"Rearranging the Timing Analysis View Layout" on page 745

- "Cross-Probing from Timing Analysis View" on page 753
- "Setting Options in Timing Analysis View" on page 733

Timing Analysis View Main Window

When you open Timing Analysis View, its main window displays the settings and preference list pane on the left. It displays the delay path information for a selected preference in the panes to the right. Each of the panes on the right can be detached from the main window, rearranged, and resized. You can use the Settings pane to view or modify the timing analysis report settings. When you select a preference from the Preference pane, you can view the path tables, schematic details, and TRACE report in the other panes. If you have enabled the Check Unconstrained Paths option in Settings, the analysis results for Unconstrained Paths will appear in the Other Reports section of the Preference pane.



Toolbar

The vertical toolbar gives you quick access to commands for exporting timing path data, changing settings, editing timing preference files (.tpf), sizing the path table columns, and displaying or hiding views.

 exports timing path delay data to a .csv file.

 allows you to edit the settings shown in the Settings pane.

 opens the Timing Preference File (TPF) version of Spreadsheet View.

 adjusts table column widths so that complete headings are visible.

 adjusts table column widths so that all of them fit inside the pane.

 toggles the display of the Path Table.

 toggles the display of the TRACE Report.

 toggles the display of the Schematic Path view.

 toggles the display of the Detailed Path Tables.

 updates the .tpf file with timing preference changes made in the TPF Spreadsheet View.

Settings

The Settings pane displays the targeted device, TRACE analysis, and report settings. You can edit a setting that is displayed in blue font. To edit a setting, right-click anywhere inside the Settings pane and choose **Settings** or click the Settings button  on the toolbar.

Settings	Values
Device Family	LatticeXP2
Device	LFXP2-17E
Package	FTBGA256
Setup Performance Grade	Default
Hold Performance Grade	Default
Check Unconstrained Connections	No
Check Unconstrained Paths	Yes
Report Asynchronous Timing Loops	No
Report Style	Verbose Timing Report
Full Name	No
Worst-Case Paths	10
Number of Unconstrained Paths	0

The following settings are included:

- Device Family – the product series name of the Lattice FPGA part, for example, LatticeXP2. This setting is read-only.
- Device **–** the product's alpha-numeric part name that appears within the software and for ordering. The name's components indicate device family, package, and device size. This setting is read-only.
- Package **–** the type of package that will house the FPGA chip and is concerned with facilitation of pin connectivity using package leads from device pads. This setting is read-only.
- Setup Performance Grade – the performance grade that will be used for setup analysis and reporting. The default is the minimum performance grade for the targeted device.
- Hold Performance Grade **–** the performance grade that will be used for hold analysis and reporting. The default is the minimum performance grade for the targeted device.

- Hold Window Performance Grade – the performance grade value of the device itself. It cannot be modified here.
- Check Unconstrained Connections – lists the paths that are not covered by any timing preference. This is a legacy option that will be discontinued after the next two Diamond releases. It is recommended that you use the Check Unconstrained Paths option instead.
- Check Unconstrained Paths – reports the paths that are not constrained, shows the start point and end point of each path, and suggests preferences to constrain the given paths.
- Report Asynchronous Timing **–** shows whether you have chosen to report asynchronous timing paths that cannot be analyzed because they are asynchronous loops.
- Report Style – shows the report style that has been selected: Verbose Report or Error Timing Report. The verbose style includes everything; whereas the error timing report only shows timing errors related to the constraints you have specified in the LPF file.
- Full Name **–** shows whether the full name setting has been selected for displaying COMP names in reports. TRACE defaults to having abbreviated COMP names in the reports, truncating names to a character limit with asterisks. The selection of Yes reports full names. This might cause some tabular formats in the ASCII file report sections to be misaligned, making reports less readable.
- Worst-Case Paths **–** the limit on the number of worst-case timing paths to be reported by TRACE. The default is 10; however, Diamond will report on and cache 200 in its memory without diminishing performance in Timing Analysis view.
- Number of Unconstrained Paths – the limit on the number of unconstrained timing paths to be reported by TRACE. Up to 200 paths can be reported. If you enter a value of 0, all unconstrained paths will be reported, up to 200.

Preferences

The Preferences pane includes two sections: Preference Reports, which shows the analysis results for all timing preferences, and Other Reports, which lists unconstrained paths that might benefit from suggested timing constraints.

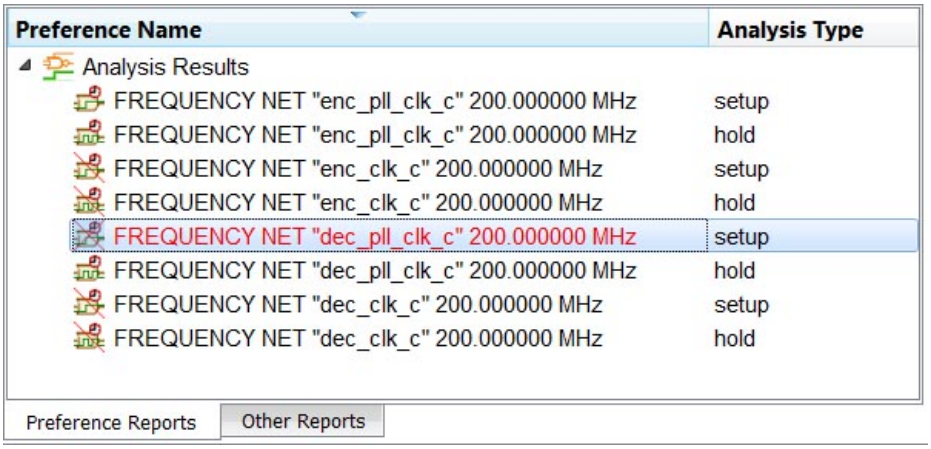
In the Preference Reports section, red font is used to highlight preferences that do not meet timing requirements. You can select a preference from the list to examine the timing paths and reports in the other panes.

The icons in front of the listed preferences have the following significance:


Delay path information is available, and the preference passes timing.

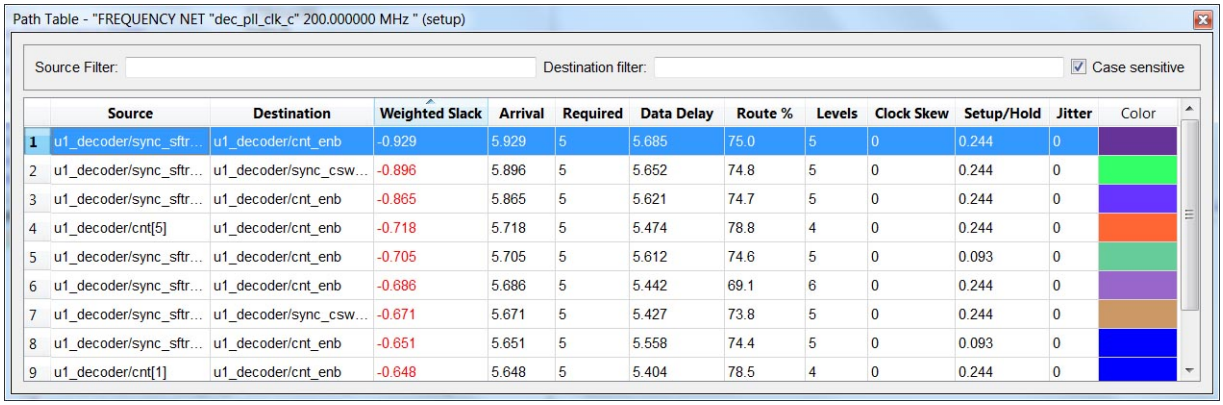

Delay path information is available, and the preference fails timing.


No delay path information is available for the preference.



Preferences Path Tables

When you select a preference from the Preferences Reports section of the Preferences pane, the path tables pane displays the available path data. For large tables, you can use the Source and Destination filters to search for items that contain a combination of different characters.



	Source	Destination	Weighted Slack	Arrival	Required	Data Delay	Route %	Levels	Clock Skew	Setup/Hold	Jitter	Color
1	u1_decoder/sync_sftr...	u1_decoder/cnt_enb	-0.929	5.929	5	5.685	75.0	5	0	0.244	0	
2	u1_decoder/sync_sftr...	u1_decoder/sync_csw...	-0.896	5.896	5	5.652	74.8	5	0	0.244	0	
3	u1_decoder/sync_sftr...	u1_decoder/cnt_enb	-0.865	5.865	5	5.621	74.7	5	0	0.244	0	
4	u1_decoder/cnt[5]	u1_decoder/cnt_enb	-0.718	5.718	5	5.474	78.8	4	0	0.244	0	
5	u1_decoder/sync_sftr...	u1_decoder/cnt_enb	-0.705	5.705	5	5.612	74.6	5	0	0.093	0	
6	u1_decoder/sync_sftr...	u1_decoder/cnt_enb	-0.686	5.686	5	5.442	69.1	6	0	0.244	0	
7	u1_decoder/sync_sftr...	u1_decoder/sync_csw...	-0.671	5.671	5	5.427	73.8	5	0	0.244	0	
8	u1_decoder/sync_sftr...	u1_decoder/cnt_enb	-0.651	5.651	5	5.558	74.4	5	0	0.093	0	
9	u1_decoder/cnt[1]	u1_decoder/cnt_enb	-0.648	5.648	5	5.404	78.5	4	0	0.244	0	

The following data is included in the path tables:

- Source – the registered cell or pad that launches data.
- Destination – the registered cell or pad that receives data.
- Weighted Slack – a measure of the amount of time that a path passes or fails the timing preference. A positive slack indicates the margin by which the path surpasses the clock preference. Negative slack indicates a failing path. The slack is “weighted” or scaled to one full clock cycle so that paths that have different cycles can be ranked according to their severity so that the most critical paths are ordered properly in the table.
- Arrival – the amount of elapsed time in arriving at the destination.
- Required – the maximum amount of time that can elapse in arriving at the destination.
- Data Delay – the total delay for the path.

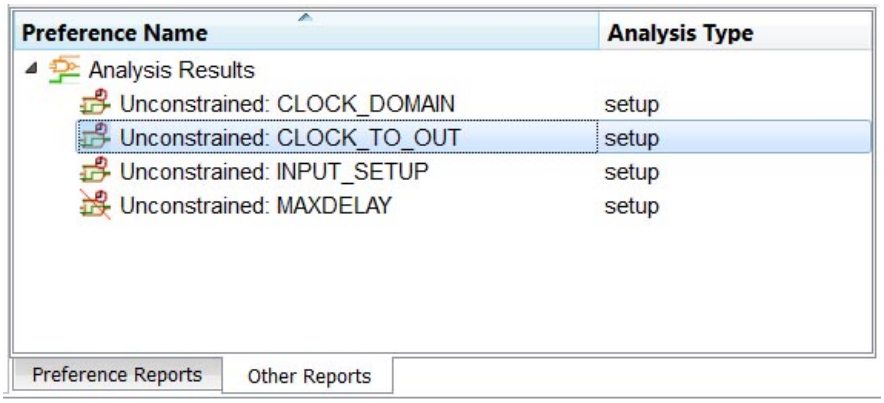
- Route **% –** the percentage of the delay that is contributed by routing resources.
- Levels – the number of combinatorial levels related to the path.
- Clock Skew – time differential between two or more destinations.
- Setup/Hold – the amount of setup and hold time required.
- Jitter – the total jitter in nanoseconds for the delay path. This is a combination of system jitter, clock jitter, and phase shift error.
- Color – how each path will be colored-coded when cross-probed to Floorplan or Physical views.

Unconstrained Paths Report

If you have selected the “Check Unconstrained Paths” option in TRACE settings, select the **Other Reports** tab in the Preferences pane to view a list of suggested constraints for unconstrained paths.

Note

The “Check Unconstrained Paths” option for Place and Route must be enabled in the strategy options or in the Timing Analysis View Settings in order for the results to appear in the Other Reports section. The Clock Path section of the Detailed Path table for unconstrained paths might be empty, since it will display only what is in the TRACE report file.

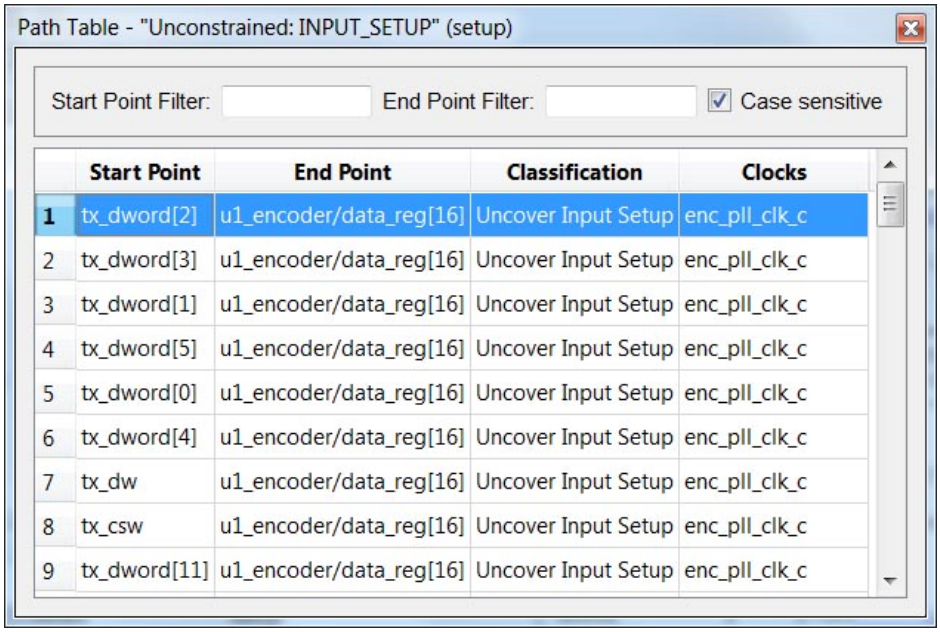


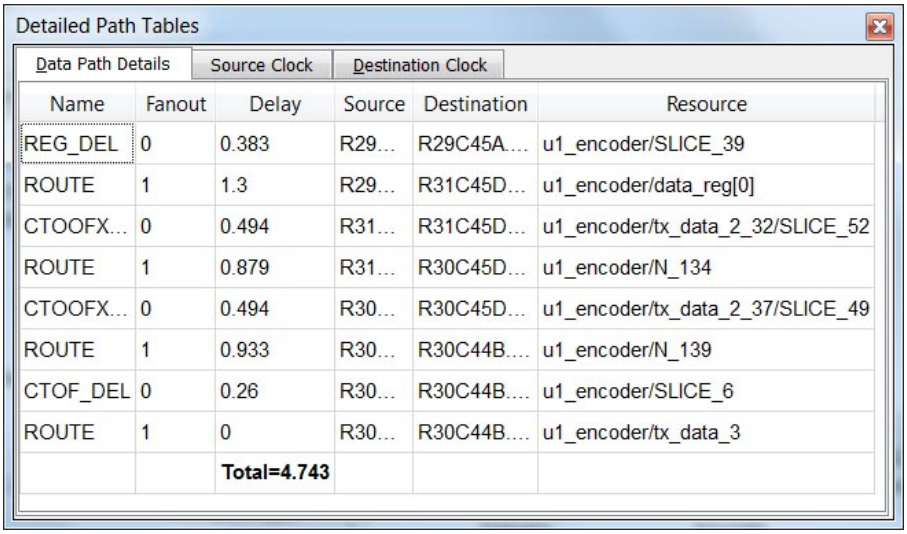
Unconstrained Paths Table

When you select a constraint from the list, the unconstrained paths are displayed in the path tables pane. The paths table shows the start point and end point of each unconstrained path and its classification. You can cross-probe a selected unconstrained path to Floorplan View, Physical View, or the TRACE report.

Detailed Path Tables

The Detailed Path Tables enable you to examine details of a selected delay path for a given timing preference. Depending on the type of preference, you can use the tabs at the top of the table to view the information by Data Path Details and Clock Path or by Data Path Details, Source Clock, and Destination clock.



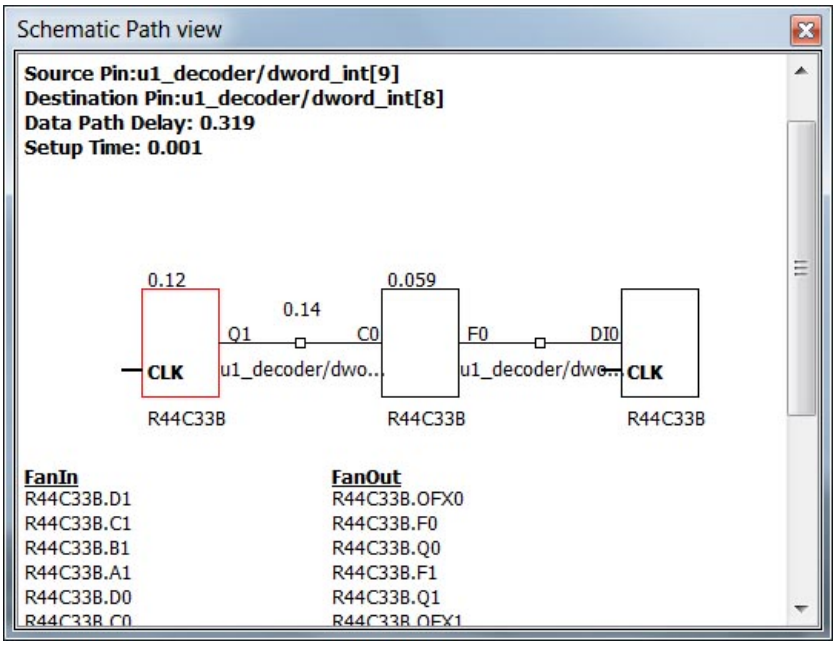


The following data is included:

- Name – the unique path identifier.
- Fanout – the number of inputs an output logic gate of the same type has in the design.
- Delay – the time related to the path in nanoseconds. This value is typically a composite of multiple timing arcs including data and clock arrival times.
- Source – the label of the device site location that contains the path's source registered cell.
- Destination – the label of the device site containing the path's destination registered cell.
- Resource – the component or net resource.

Schematic Path View

The Schematic Path View displays a graphical image of the timing path. Each component is represented by a large square box, and the location is shown beneath the box. Each net is represented by a small square. Double-click a component box to view the fanin and fanout.



Report View

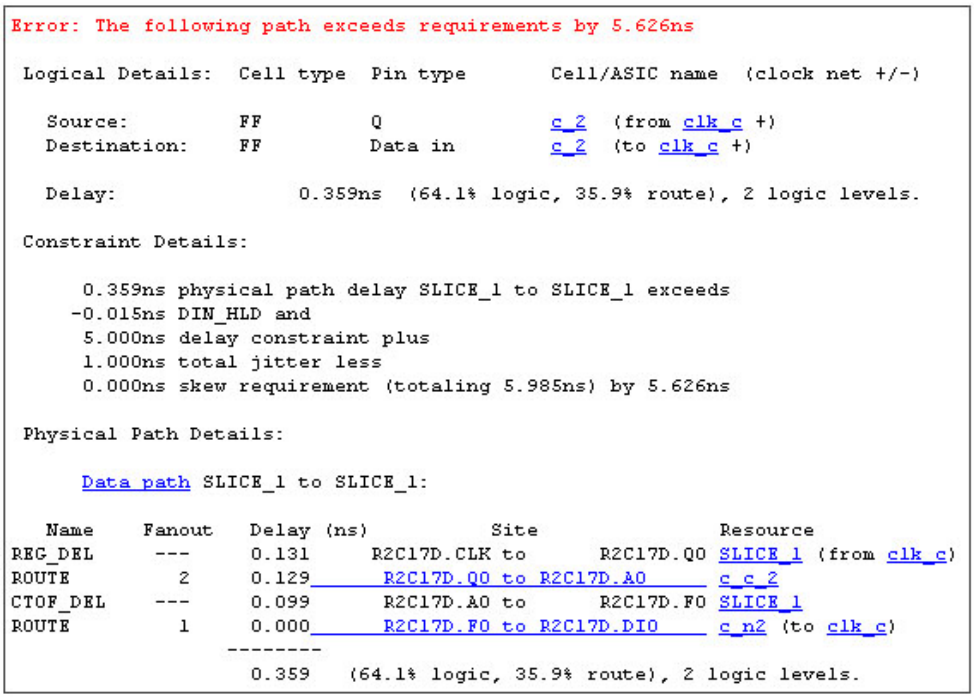
The Report View presents the timing analysis report in a text format and shows each preference that is applied by the timing analysis program TRACE.

You can access a timing preference section in the report by right-clicking a timing preference from the Preference Name pane and choosing **Show in Report**. You can access timing path information by right-clicking a path from the Path Tables pane at the top and choose **Show In > Report**

If a timing error is present in the report, the preference will appear in red font in the Preference Name listing and also in the TRACE report in the Report pane.

Blue font indicates a hyperlinked component name or net name. Right-click the blue text to access the “Show In” command. This allows you to cross-probe the component or net to Floorplan View or Physical View.

See Also
- “Rearranging the Timing Analysis View Layout” on page 745
- “Cross-Probing from Timing Analysis View” on page 753
- “Setting TRACE Options” on page 696



```
Error: The following path exceeds requirements by 5.626ns

 Logical Details:  Cell type  Pin type       Cell/ASIC name  (clock net +/-)

   Source:         FF         Q              c_2  (from clk_c +)
   Destination:    FF         Data in        c_2  (to clk_c +)

   Delay:               0.359ns  (64.1% logic, 35.9% route), 2 logic levels.

 Constraint Details:

      0.359ns physical path delay SLICE_1 to SLICE_1 exceeds
     -0.015ns DIN_HLD and
      5.000ns delay constraint plus
      1.000ns total jitter less
      0.000ns skew requirement (totaling 5.985ns) by 5.626ns

 Physical Path Details:

      Data path SLICE_1 to SLICE_1:

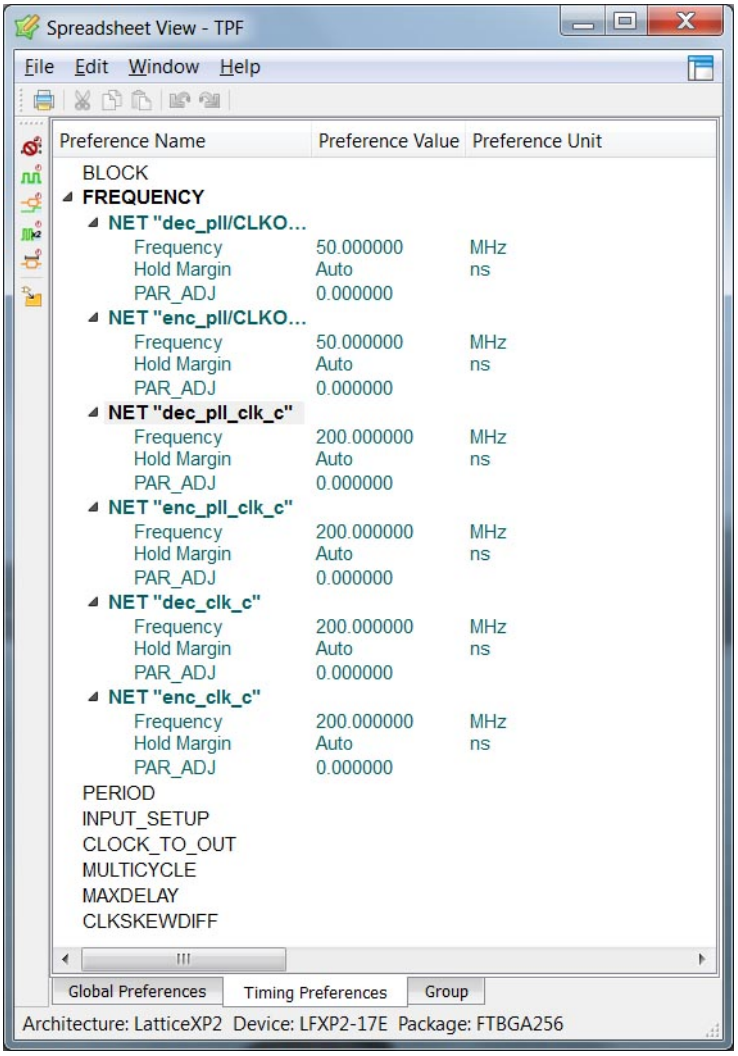
   Name     Fanout   Delay (ns)          Site               Resource
REG_DEL      ---     0.131     R2C17D.CLK to      R2C17D.Q0 SLICE_1 (from clk_c)
ROUTE          2     0.129      R2C17D.Q0 to R2C17D.A0      c_c_2
CTOF_DEL     ---     0.099      R2C17D.A0 to      R2C17D.F0 SLICE_1
ROUTE          1     0.000      R2C17D.F0 to R2C17D.DI0     c_n2 (to clk_c)
                   --------
                     0.359   (64.1% logic, 35.9% route), 2 logic levels.
```

Spreadsheet View – TPF

The Timing Preference File (TPF) version of Spreadsheet View contains three preference sheets that allow you to modify timing preferences: Global Preferences, Timing Preferences, and Group. It enables you to modify or create new timing preferences, such as PERIOD or FREQUENCY; global preferences such as voltage and Block Path; and new port, cell, or ASIC groups. The interface, including the dialog boxes, is very similar to the regular Spreadsheet View. However, when you return to the Timing Analysis View to save your timing preference changes, the changes are written to a timing preference file (.tpf) instead of the active logical preference file (.lpf). The .tpf file is then added to the Analysis Files folder in Diamond's File List pane.

The TPF Spreadsheet View can be detached and reattached in the same manner as the Timing Analysis View main window.

See Also
- "Modifying Preferences for Timing Analysis" on page 734
- "Importing In-Memory Timing Preferences" on page 736
- "Creating a New Timing Preference File" on page 747
- "Exporting Timing Preferences to a Logical Preference File" on page 751



Rearranging the Timing Analysis View Layout

Timing Analysis View's flexible interface enables you to detach the different panes, rearrange them, and resize them to make the timing details easier to view.

Detaching or Reattaching Timing Analysis View You can detach Timing Analysis View from Diamond's main window to give you more room to examine timing data and rearrange the layout of panes.

To detach Timing Analysis View:

- Click the Detach Tool button at the top right of the Timing Analysis View main window..

  This detach button is represented by a dim graphic, which distinguishes it from the detach button of the individual panes.

To reattach Timing Analysis View:

- Choose **Window >** **Attach Window**.

Detaching or Reattaching a Pane Each pane of Timing Analysis View, except the Settings pane and the Preferences pane, can be detached as a separate view or repositioned to a different area of the Timing Analysis View window.

To detach a pane, do one of the following:

- Click the detach button at the top right of the pane.
- Click and hold the pane's title bar and drag it out of the Timing Analysis View window.

To reattach a pane, do one of the following:

- Click and hold the pane's title bar and drag it back to the desired position in Timing Analysis View.
- Double-click the pane's title bar to attach it back into the Timing Analysis View window.

Repositioning a Pane You can rearrange the positions of the panes inside or outside of the Timing Analysis View window.

To reposition a pane inside or outside of Timing Analysis View:

- Click and hold the pane's title bar and drag it to the desired position.

  If you are repositioning a pane inside the Timing Analysis View window, it will snap into place when you release the mouse button.

Resizing a Pane You can resize any of the panes of Timing Analysis View, including the Settings and Preferences panes.

To resize a pane that is inside the Timing Analysis View window:

1. Hold your mouse over the vertical or horizontal border that separates the pane from the neighboring pane.
2. When you see the arrows and lines symbol, hold the mouse down and drag the border to the desired width or height.

To resize a pane that is outside the Timing Analysis View window:

- Hold down any border or corner and drag the pane to the desired size.

See Also "Timing Analysis View Main Window" on page 737

- "Cross-Probing from Timing Analysis View" on page 753

Working with Timing Preference Files

Timing preference files (.tpf) are files that you create in order to experiment with timing preferences without having to rerun Place & Route. When you first open Diamond to begin a new project implementation, you have no timing preference files (.tpf) associated with your project. A default Untitled.tpf file is auto-generated that uses constraints from your project's HDL source or logical preference (.lpf). To save any changes to your timing preferences, you must use the Save As command and save the Untitled file with a different name.

One way to create your first .tpf file is to modify timing preferences using the TPF Spreadsheet View, and then return to the Timing Analysis View main window and save the changes using the Save As command. After you have set the new .tpf file as the active one, you can simply use the Save command to save further changes.

You can create multiple sets of timing preference files for your project. They will appear in the Analysis Files folder of the File List pane. The .tpf files are not used directly by Diamond as input into your project implementation; instead, they are used for experimentation and analysis. You can later copy and paste the preferences from the .tpf file into your logical preference file (.lpf). You can also export a selected timing preference from the TPF Spreadsheet View to the .lpf file.

See Also ▶"Modifying Preferences for Timing Analysis" on page 734

Creating a New Timing Preference File

You can create a new timing preference file (.tpf) after placement and routing by modifying preferences in the TPF Spreadsheet View and saving the changes to a new .tpf file. You can also create a new .tpf file at any time by using the file commands in Diamond's main window.

To create a new .tpf file from preferences in TPF Spreadsheet View:

1. In the TPF Spreadsheet View, modify timing preferences or create new ones. See "Modifying Preferences for Timing Analysis" on page 734.
2. Return to the Timing Analysis View main window.

   The tabs of Timing Analysis View and Spreadsheet View – TPF are now marked with asterisks, indicating that changes have been made. If a .tpf file has not already been loaded, the title bar of Timing Analysis View shows "Untitled *." The Update button on the vertical toolbar is rotating.

   Note

   If desired, click the **Update** button to immediately view the results of the changes.

3. In Timing Analysis View, choose **File > Save Untitled As** and specify a name and location for the .tpf file.
4. Click **Save**.

The file name and path of the new .tpf file are displayed in the title bar of Timing Analysis View. The new .tpf file is listed in the Analysis Files folder of the File List pane.

To create a new .tpf file from Diamond's main window:

1. Choose **File > New >**  **File**.

   Optionally, right-click the Analysis Files folder in the File List pane and choose **Add > New File**.

2. In the dialog box, select **Source Files** from the Categories list, and then select **Timing Preference File** from the Source Files list.
3. Type a name in the Name box and browse to the desired location and click **OK**.
4. To add the .tpf file to the active implementation, make sure that the **Add to Implementation** option is selected.

   Note

   If you clear this option, the new .tpf file will not appear in the Analysis Files folder in the File List pane.

5. Click **New**.

   Diamond opens Timing Analysis View if it is not already open. The new .tpf file name and path appear in the title bar. If you selected the "Add to Implementation" option, the .tpf file appears in the Analysis Files folder of the File List pane.

See Also
- "Opening a Timing Preference File in Timing Analysis View" on page 750
- "Setting a Timing Preference File as the Active File" on page 749
- "Adding a Timing Preference File to the Analysis Files Folder" on page 748

Adding a Timing Preference File to the Analysis Files Folder

If you have created timing preference files (.tpf) without selecting the "Add to Implementation" option, they will not appear in the Analysis Files folder in the File List pane. Adding .tpf files to the Analysis Files folder gives you immediate access to them in Timing Analysis View and enables you to select one of them as the active .tpf file.

To add a .tpf file to the Analysis Files folder:

1. Right-click the **Analysis Files** folder and choose **Add > Existing File** from the pop-up menu.
2. In the Add Existing File dialog box, navigate to the .tpf file that you want to add, select it, and choose **Add**.

The .tpf file you selected is added to the Analysis Files folder.

See Also ▶"Creating a New Timing Preference File" on page 747

- "Removing a Timing Preference File from the Analysis Files Folder" on page 749

Removing a Timing Preference File from the Analysis Files Folder

When you remove a .tpf file from the Analysis Files folder, it is only deleted from the list. The file remains in your project directory or the folder where you saved it, and you can always add it back to the Analysis Files list.

To remove a .tpf file from the Analysis FIles Folder:

- Expand the Analysis Files folder, right-click the .tpf file that you want to remove, and choose **Remove**.

See Also ▶"Creating a New Timing Preference File" on page 747

- "Adding a Timing Preference File to the Analysis Files Folder" on page 748

Setting a Timing Preference File as the Active File

A timing preference file (.tpf) that has been designated as active will be loaded automatically when you open Timing Analysis View. Your active .tpf file is the one displayed in bold font in the Analysis Files folder of the File List pane.

To set a .tpf file as the active file:

1. In Diamond, select the File List tab, and then expand the Analysis Files folder.
2. Right-click the .tpf file that you wish to designate as the active one and choose **Set as Active TPF** from the pop-up menu.

   The .tpf file that you selected now appears in bold font, indicating that it is the active one. The next time you open Timing Analysis View, it will be loaded automatically.

   If Timing Analysis View is already open with a different .tpf file, the newly activated .tpf file will not be loaded automatically. In this case, you must reopen Timing Analysis View in order to load the active .tpf file. You can do this in one of the two following ways:

   - Close Timing Analysis View manually and reopen it.
   - Right-click the active .tpf file in the Analysis Files folder, choose **Open With > Timing Analysis View**, and click **OK**.

See Also ▶"Creating a New Timing Preference File" on page 747

- "Opening a Timing Preference File in Timing Analysis View" on page 750
- "Adding a Timing Preference File to the Analysis Files Folder" on page 748

Opening a Timing Preference File in Timing Analysis View

After creating one or more timing preference files (.tpf) you can open one of them in Timing Analysis View. You can open a .tpf file from Diamond's File List pane or from the File menu of Timing Analysis View if it is detached.

To open a .tpf file in Timing Analysis View from Diamond's File List pane:

1. In the File List pane, expand the Analysis Files folder.
2. Right-click the .tpf file that you want to open and choose **Open With** from the pop-up menu.
3. In the Open With dialog box, select **Timing Analysis View**.
4. If you want to set Timing Analysis View as the default for opening and viewing a .tpf file, click **Set as Default**.

   This will enable you to double-click a .tpf file from the Analysis Files folder to open it in Timing Analysis View instead of the source editor.
5. Click **OK**.

   Diamond opens Timing Analysis View if it is not already open, and it loads the .tpf file that you selected.

To open a .tpf file from within Timing Analysis View:

1. In Timing Analysis View, click the Detach Tool button  .in the upper right corner.
2. From the detached Timing Analysis View, choose **File > Open File**.
3. In the Open Timing Preference File dialog box, select the desired .tpf file and click **Open**.

See Also "Creating a New Timing Preference File" on page 747

- "Adding a Timing Preference File to the Analysis Files Folder" on page 748
- "Opening a Timing Preference File in the Source Editor" on page 750

Opening a Timing Preference File in the Source Editor

Although you can only open one timing preference file (.tpf) at a time in Timing Analysis View, you can have multiple .tpf files open at the same time in Diamond's Source Editor. Multiple .tpf files that are open in the Source Editor are arranged in tabs at the top of the window, allowing you to make quick

comparisons of timing preference usage. Timing preferences that you edit in the Source Editor will be reflected when you next load the .tpf file into Timing Analysis View.

You can use the Analysis Files folder or Diamond's File menu to open a .tpf file in the Source Editor.

To open a Timing Preference File in the Source Editor using the Analysis Files folder:

1. From the Analysis Files folder, right-click the desired .tpf file and choose **Open With** from the pop-up menu.
2. In the Open With dialog box, select **Source Editor**.
3. If you want to set the Source Editor as the default for opening and viewing a .tpf file, click **Set as Default**.

   This will enable you to double-click a .tpf file from the Analysis Files folder to open it in the Source Editor instead of Timing Analysis View.
4. Click **OK**.

   The file opens as a text file in the Source Editor window, and the name of the .tpf file is displayed on the tab.

To open a Timing Preference File in the Source Editor using Diamond's File menu:

1. In Diamond, choose **File > Open > File**.
2. In the Open File dialog box, choose **Timing Preference File (*.tpf)** from the Files of type drop-down menu.
3. Select the desired .tpf file and click **Open**.

   The file opens as a text file in the Source Editor window, and the name of the .tpf file is displayed on the tab.

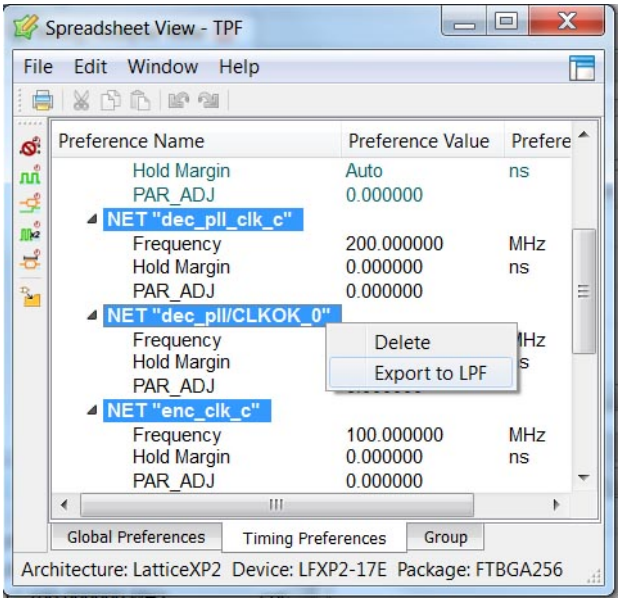
See Also
- "Creating a New Timing Preference File" on page 747
- "Adding a Timing Preference File to the Analysis Files Folder" on page 748

Exporting Timing Preferences to a Logical Preference File

After experimenting with timing preferences in the Timing Analysis view, you can use the TPF Spreadsheet View to export some of the preferences from the timing preference file (.tpf) to a logical preference file (.lpf). You can do this with preferences that are on the Timing Preferences sheet or the Group sheet.

To export timing preferences from the .tpf file to the .lpf file:

1. In Timing Analysis view, click the **Change timing preferences** button to open the TPF Spreadsheet View.
2. Select the Timing Preferences sheet or the Group sheet.
3. Select the preferences that you want to export. Hold down the Ctrl key to select multiple preferences.



4. Right-click the selected preferences and choose **Export to LPF**.

   The regular Spreadsheet View opens. Its tab contains an asterisk, indicating that unsaved preferences are in memory. The Timing Preferences or Group sheet shows the preference changes that you exported.

5. Save the changes to the currently active .lpf file using the **Save** command, or choose **File > Save As** to save the changes to a different .lpf file.

Note

To implement the preference changes to the currently active .lpf file, you must rerun your design flow through Place & Route Design.

See Also
- "Modifying Preferences for Timing Analysis" on page 734
- "Creating a New Timing Preference File" on page 747
- "Copying Timing Preferences to a Logical Preference File" on page 753

Copying Timing Preferences to a Logical Preference File

You can copy and paste timing preferences from the .tpf file to a .lpf file using the Source Editor, or you can save the entire .tpf file as an .lpf file.

To copy timing preferences from the .tpf file to the .lpf file:

1. Open the desired .tpf and .lpf files in the Source Editor.
2. In the .tpf file, select the timing preference text that you want to copy and choose **Edit > Copy**.
3. In the .lpf file, choose **Edit > Paste**, and then choose **File > Save <file_name>.lpf.**

To save the entire .tpf file as an .lpf file:

1. Open the .tpf file in the Source Editor.
2. Choose **File > Save <file_name>.tpf As** and in the Save as type box, choose **Preference Files**.
3. Navigate to the directory where you want to save the .lpf file.
4. In the File name box, type a file name and change the .tpf extension to .lpf.
5. Click **Save**.

See Also ▶"Opening a Timing Preference File in the Source Editor" on page 750

▶ "Exporting Timing Preferences to a Logical Preference File" on page 751

Cross-Probing from Timing Analysis View

In Diamond, many views contain corresponding cross-reference data that allow you to "jump" between views. This cross-probing feature allows you to gain new perspectives on your layout and connectivity for particular design elements that are critical for meeting your constraints. This can be very useful in guiding your design run iterations toward specific goals.

Timing Analysis View enables cross-probing within the view, and it allows you to cross-probe selected elements to Floorplan View and Physical View. You can also use this feature to group components along an individual signal. See "Grouping Components Along an Individual Signal" on page 515.

You can cross-probe an unconstrained path as well as a path that is constrained by a defined preference.

To cross probe from Timing Analysis View:

1. In the Preferences pane, do one of the following:
   - Select the Preferences Reports tab, and then select a defined timing preference. For example, you might have created setup and hold time analysis with a FREQUENCY preference for a clock in your design.
   - Select the Other Reports tab, and then select a preference for unconstrained paths.

     The Other Reports lists each unconstrained path by suggested preference and shows the analysis type. When you select a preference from the list of unconstrained paths, the Path table shows the start point, end point, and clocks. It also shows the classification: register-to-register, input-setup, clock-to-out, or port-to-port.

   The Path Table and the Detailed Path Table become populated with various data on delay and timing related to that preference.

   Note

   If you have not set any timing preferences, see "Modifying Preferences for Timing Analysis" on page 734.

2. Do one or more of the following:
   - Right-click a timing preference in the Preference Name pane and choose **Show in Report**.

     The preference is immediately highlighted in the Report pane.
   - Right-click a path from the Path Table and choose **Show In > Floorplan View**.

     Floorplan View opens with a zoomed-in view of the path. The path is highlighted with the same color shown in the Path Table for the selected delay path.
   - Right-click a path from the Path Table and choose **Show In > Physical View**.

     Physical View opens with a zoomed-in view of the path. The path is not differentiated in color according to the Path Table, but is highlighted with the default color for delay paths.
   - Right-click a path from the Path Table and choose **Show In > Report**.

     The Report pane jumps to the path section and displays the path in blue.
   - In the Reports pane, right-click an element that is highlighted in blue and choose **Show In > Floorplan View** or **Show In > Physical View**.

     The path is highlighted in the layout view you selected.

See Also "Timing Analysis View Main Window" on page 737

Exporting Timing Path Delay Data from Timing Analysis View

The path delay data displayed in the Path and Detailed Path tables of Timing Analysis view can be exported to a comma-separated value file (.csv). This enables you to view the data in an external spreadsheet program.

To export timing path delay data to a .csv file from Timing Analysis view:

1. In Timing Analysis View, select a timing preference from the Preference Name pane.

   The Path Table and Detailed Path Table should automatically be populated with timing data associated with the preference.

2. Choose **File > Export** or click the Export button on the vertical toolbar.

3. In the Export dialog box, type a name in the File Name text box, browse to the desired location, and click **Save**.

   The .csv file is saved in the location you specified.

See Also

- "Timing Analysis View Main Window" on page 737
- "Modifying Preferences for Timing Analysis" on page 734
- "Exporting Timing Preferences to a Logical Preference File" on page 751

Chapter 10

Analyzing Power Consumption

Included with the Diamond software is Power Calculator, which estimates the power dissipation for a given design. Power Calculator uses parameters such as voltage, temperature, process variations, air flow, heat sink, resource utilization, activity, and frequency to calculate the device power consumption. It reports both static and dynamic power consumption.

Power Calculator allows you to import frequencies from the post-PAR trace report file (.twr file), and it allows you to import activity factors from the post-PAR simulation file (.vcd file). After the design information is added, Power Calculator provides accurate power consumption analysis for the design.

Power Calculator provides two modes for reporting power consumption: estimation mode, which can be used before completing the design; and calculation mode, which is based on the physical netlist file (.ncd) after placement and routing.

You can open Power Calculator from within Diamond or as a stand-alone tool from the Windows Start menu. Either method provides both estimation mode and calculation mode functionality. A stand-alone Power Estimator is also available, which allows you to estimate power consumption without having Lattice Diamond installed.

Calculation Mode Within the Diamond environment, Power Calculator calculates power consumption based on the project's files, including device information. This information is automatically extracted when you open Power Calculator from within Diamond. In the stand-alone Power Calculator, power consumption is calculated based on a selected native circuit description (.ncd) file or power calculator file (.pcf)

Estimation Mode Within the stand-alone Power Calculator or Power Estimator, the Startup Wizard enables you to estimate power based on a selected device, and it gives you the option of including a template of resource settings. Within the Diamond environment, Power Calculator will estimate power for a unrouted design. After the design is routed, it enables you to change the device family or other data and obtain the estimated power consumption.

See Also ▶"Software Mode" on page 762

- "Starting Power Calculator from Diamond" on page 757
- "Starting Power Calculator as a Stand-Alone Tool" on page 757

- "Power Analysis Design Flow" on page 759

Starting Power Calculator from Diamond

Power Calculator is available from Diamond as soon as you have opened a project.

To start Power Calculator from the Diamond Tools menu or toolbar:

1. In Diamond, open a project or create a new one.
2. Choose **Tools > Power Calculator** or click the  button on the toolbar.

   Power Calculator opens in estimation mode or calculation mode, depending on the design stage, and displays the Power Summary page.

   When Power Calculator opens in calculation mode, it imports any Bank VCCIO settings that are in the preference file and displays these settings on the I/O page.

You can also start Power Calculator by creating or opening a new Power Calculator file (.pcf):

- Use the File menu or the Analysis Files folder to create a new Power Calculator file (.pcf).

  Power Calculator opens and loads the newly created .pcf file.

- Use the File menu or the Analysis Files folder to open an existing .pcf file.

  Power Calculator opens in estimation mode or calculation mode, depending on the status of the selected .pcf file. If it opens in calculation mode, it will import any Bank VCCIO settings that are in the preference file and display these settings on the I/O page.

See Also
- "Running Power Calculator from the Tcl Console" on page 759
- "Saving a Power Calculator File" on page 773
- "Adding Power Calculator Files to the Analysis Files Folder" on page 776
- "Setting a Power Calculator File as the Active Analysis File" on page 777

Starting Power Calculator as a Stand-Alone Tool

Power Calculator is available as a stand-alone tool from the Windows Start menu. This allows you to work with a Power Calculator project, in calculation mode or in estimation mode, without opening Diamond. The Startup Wizard enables you to create a new Power Calculator project, based on a selected device or a processed design, or to open an existing Power Calculator project file (.pcf).

To start Power Calculator as a stand-alone tool:

1. Choose **Start > Programs > Lattice Diamond > Accessories > Power Calculator**.

   After a few moments, the Power Calculator Startup Wizard appears.

2. Do one of the following, and then click **OK**:

   - To calculate power consumption based on a mapped or placed and routed design, select **Calculate power with design (NCD)**.

     Click the NCD File Browse button to navigate to the .ncd file.

     After you select the .ncd file, the File Name and File Directory boxes are automatically filled in by the Startup Wizard. If you want to give the Power Calculator project file a different name than the .ncd file name, type the name in the File Name box.

     When you click OK, Power Calculator opens in calculation mode. If any VCCIO bank preferences have been set in the preference file, they will be imported and displayed on the I/O page.

   - To estimate power consumption based on a selected device, select **Estimate power with device selection**.

     The device selection menus appear at the bottom of the Startup Wizard.

     Type a name for the Power Calculator project file in the File Name box and browse to the desired directory for the project. Make your selections from the device menus.

     If you would like to use a template of resource settings, select the **Use Template** option, click **Select**, and do one of the following:

     - Select **Specify Resource by Design Type**, and then select the design type from the drop-down menu and click **OK**.
     - Select **Specify Resource by Component Utilization**, and then select from the options provided and click **OK**.

       The Use Template text box displays the resource settings or the design type that you selected.

     When you click OK, Power Calculator opens in estimation mode and loads the device settings you specified.

   - To open an existing Power Calculator project, select **Open existing PCF file**.

     Click the Browse button to navigate to the .pcf file.

     When you click OK, Power Calculator opens in estimation mode or calculation mode, depending on the status of the .pcf file.

     If Power Calculator opens in calculation mode, it will import any Bank VCCIO settings that are in the preference file and display these settings on the I/O page.

See Also "Running Power Calculator from the Tcl Console" on page 759

"Saving a Power Calculator File" on page 773

Running Power Calculator from the Tcl Console

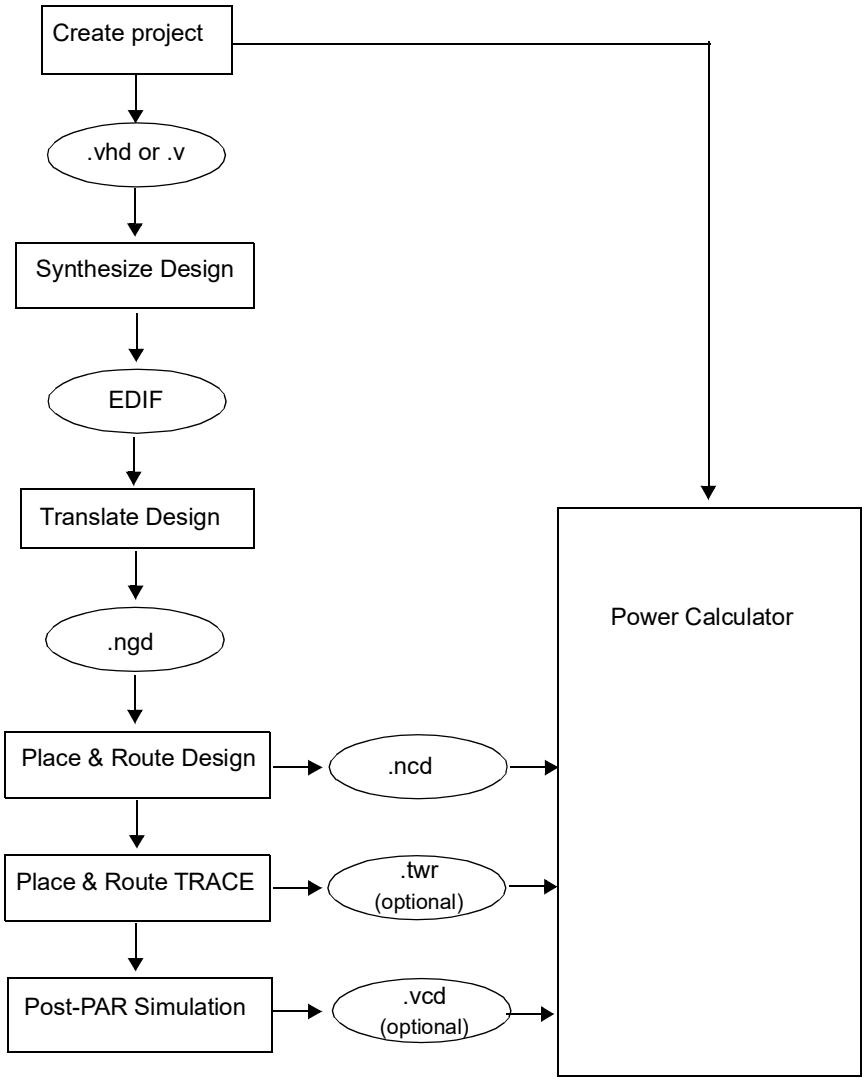
Diamond's Tcl console window enables you to use Tcl commands to perform many power analysis functions. For a complete list and descriptions of Power Calculator Tcl commands, see "Power Calculator Tcl Commands" on page 2615 in the Tcl Command Reference Guide.

See Also ▶"Tcl Command Reference Guide" on page 2586

Power Analysis Design Flow

Power Calculator supports all Lattice FPGA devices. The design flow involved in using Power Calculator in Diamond is as follows.

Figure 8:



See Also ▶"Inputs" on page 760

▶ "Outputs" on page 760

Inputs

When you first launch Power Calculator from Diamond, it displays information from your design project. For an unrouted design, it shows default resource information based on the targeted device. For a routed design, it extracts information from the placed and routed .ncd file.

Additionally, Power Calculator accepts the following files as optional input.

- Value change dump file, <*project_name*>.vcd, which is an ASCII file containing activity and frequency information. Its format is specified by the IEEE 1364 standard. It should be in the format of gate-level simulation and match the design. The .vcd file preserves waveform information that can be displayed in third-party tools such as Active-HDL.

  If you provide a post-routed simulation .vcd file, Power Calculator looks up the clock signals in the .vcd file and compares them to each clock in the Power Calculator pages. If the clock names match, Power Calculator takes the frequency data from the .vcd file and populates the frequency columns, the activity factor (AF(%)) columns, or both, in the pages that contain these columns.

- Trace report file, <*project_name*>.twr, which is an FPGA timing report file produced by the Place & Route Trace process in Diamond. The .twr report file enables you to determine to what extent the timing constraints for a design have been met. When you import this file, Power Calculator obtains from it the frequencies with which to populate the frequency columns. To obtain frequencies from a .twr file, you must have a frequency constraint on the clock net or the clock port; otherwise, no frequencies will be entered into Power Calculator. You can set the frequency constraint in Spreadsheet View's PERIOD/FREQUENCY Preference dialog box.

See Also ▶"Importing a Value Change Dump (.vcd) File" on page 783

▶ "Changing the Global Default Activity Factor" on page 783

Outputs

Power Calculator generates power calculation results in tabular format from information extracted from the design project. All of this information, when saved, is kept in the project's Power Calculator file (.pcf).

Power Calculator also generates a power calculation report, which can be viewed in HTML or text format. It also generates power graphs that show how power consumption is affected with varying voltage, temperature, and clock frequency.

See Also ▶"Saving a Power Calculator File" on page 773

- "Generating Power Graphs" on page 795
- "Viewing the Power Calculator Report" on page 797

Static and Dynamic Power Consumption

Power Calculator reports the dynamic and static portion of the power dissipation. Power refers to the power consumed by the design. It is based on the extracted data from a placed and routed design file (.ncd) or on the estimation information that you provide.

The dynamic portion is the power consumed by the used resources while they are switching. The power dissipation of the dynamic portion is directly proportional to the frequency at which the resource is running and the number of resource units used.

The static portion of power consumption is the total power consumed by the used and unused resources.

Activity Factor Calculation

To calculate the power consumption for the routing interconnect, logic, and the read/write ports in an embedded block, Power Calculator requires the frequency and an activity factor percentage. The activity factor percentage is the percentage of time that a registered output node is active relative to a specified clock. Most of the resources associated with a clock domain run or toggle at a percentage of the frequency at which the clock runs.

The frequency appears as the "Freq. (MHz)" column on most pages. The activity factor percentage appears as the "AF (%)" column.

See Also

- "Importing a Value Change Dump (.vcd) File" on page 783
- "Changing the Global Default Activity Factor" on page 783
- "Changing the Global Default Frequency Setting" on page 784

Power Calculator Window Features

Power Calculator's main window displays the software mode being used and the currently selected page of power consumption information. When you first open Power Calculator, the Power Summary page appears by default. All other pages of information for the device are made available from the tabs arranged at the top of the main window. This section describes these features and the color coding of cells.

Software Mode

The Software Mode in the top right corner of the Power Calculator window indicates whether Power Calculator is running in estimation or calculation mode. This field is read-only.

Estimation Mode In estimation mode, Power Calculator provides estimates of power consumption based on the device resources or template that you provide. This mode enables you to estimate the power consumption for your design before the design is complete or even started. It is useful for “what if” analysis. You must supply the frequency, activity factor, and voltage.

Calculation Mode In calculation mode, Power Calculator calculates power consumption on the basis of device resources taken from a design’s .ncd file, or from an external file such as a value change dump (.vcd) file, after placement and routing. This mode is intended for accurate calculation of power consumption, because it is based on the actual device utilization.

Reverting to Estimation Mode Power Calculator will revert to estimation mode from calculation mode in the following circumstances:

- If you start using Power Calculator in calculation mode and change the data in any cell other than AF (%), Freq., V., Dynamic Power Multiplier, Ambient Temperature, Performance Grade, Operating Condition, or Process Type, Power Calculator will automatically revert to estimation mode.
- Rerunning a process will change the software mode if you are working with the unsaved “untitled” temporary power calculator file. For example, if you rerun Map Design while in calculation mode using the “untitled” file, the tool will revert to estimation mode. If you are working with a saved .pcf file, the software mode will not change when you rerun a process.

Reverting to Calculation Mode For many types of changes, Power Calculator enables you to revert to calculation mode from estimation mode, if you have not yet saved the changes to the .pcf file. See “Reverting to Calculation Mode” on page 782 for more information.

Power Summary

The Power Summary summarizes the conditions of power consumption. It is the first page to open when you run Power Calculator. The Power Summary enables you to change the targeted device, operating conditions, voltage, and other basic parameters. Updated estimates of power consumption are then displayed based on these changes.

Device The Device section enables you to select a device family, package, part name, performance grade, and operating conditions. It displays power information based on these selections.

You can specify one of the following operating conditions, depending on the device:

- Industrial – Devices are rated at 105 degrees Celsius.

- Commercial – Devices are rated at 85 degrees Celsius.
- Automotive – Devices are rated at 125 degrees Celsius.

Device Power Parameters Section The Device Power Parameters section contains information pertaining to the device process conditions and power model status.

The process type specifies the corners or conditions under which the device was manufactured. It can be one of the following:

- Typical – to reflect the typical amount of current consumed by the circuit.
- Worst – to reflect the maximum amount of current consumed by the circuit.

The Power File Revision displays the status of the model:

- Advanced – The power file has all the constants, functions, formulas, and defaults. The constants are based on silicon simulation data without extracted layout parameters. The status of this file is “advanced.”
- Preliminary – The power file has all the constants, functions, formulas, and defaults. The constants are based on nominal silicon characterization data rather than simulation data. The status of this file is “preliminary.”
- Final/Production – The power file has all the constants, functions, formulas, and defaults. The constants are based on silicon characterization data that includes characterization of corner lots and conditions rather than simulation data. The status of this file is “production.”

Device Power Saving Selection Section The Device Power Saving Selection section, available for LIFMD/F devices, allows you to select Normal mode or Sleep mode from the drop-down menu. This affects the coefficients used in calculations.

Power Control The Power Control button is available for MachXO2, MachXO3D, MachXO3L, ECP5, LIFMD/F, and Platform Manager 2 devices. It opens the Power Option Controller, which enables you to dynamically change the power of certain blocks through the use of standby mode and other options. See “Controlling Power Options for Low-Power Devices” on page 790 for more information.

Environment Section The Environment section displays information about the operating temperature of the device. The Thermal Profile button enables you to choose the thermal impedance model for the power-consumption or current-consumption calculation. The Ambient Temperature cell allows you to specify the surroundings at which the device is expected to operate, in degrees Celsius. Temperature values must be between -40 and +125 degrees Celsius.

Effective Theta-JA specifies the cumulative thermal impedances of a particular system. This figure is used in calculating the junction temperature (Tj) of a die in a particular environment according to the following formula:

Tj = *power* * *theta_effective* + *ambient_temperature*

Junction Temperature specifies the temperature of the die within the device package, in degrees Celsius. You can adjust the junction temperature by using the models in the Thermal Profile dialog box. If the calculated value in the Junction Temperature box is either larger than 125 degrees Celsius or larger than the maximum junction temperature allowed for a particular device and operating conditions, a red background appears in the Junction Temperature box.

Maximum Safe Ambient specifies the maximum safe operating temperature for a die. If this temperature is exceeded, the semiconductor physics of the silicon change, so the die can operate erratically. The life of the device is also shortened.

Voltage/Dynamic Power Multiplier This section contains information about the estimated power or current consumption by power supply. The Dynamic Power Multiplier specifies the derating factor for the dynamic portion of the power consumption. Power Calculator does not include derating for dynamic power. This factor enables you to change the number by the derating factor that you want to apply. The derating factor specified must be between 0.5 and 2. The default is 1.0.

Current by Power Supply This section displays the static, dynamic and total current consumption in amperes.

Power by Power Supply This section displays the static, dynamic, and total power consumption in watts.

Power by Block This section displays the total power consumption by the different types of blocks, in watts.

Peak Startup (A) This section shows how much peak startup current each supply in the design draws, in amperes. When the “N/A” notation appears in this column, it means that no data is available yet. However, you can call Lattice Semiconductor Technical Support for this information.

Note

The peak startup current for LatticeECP3 devices LFE-70EA and LFE-95EA is at the commercial temperature limit of 85° Celsius.

Power Calculator Pages

Each time Power Calculator opens, it displays the Power Summary, which shows the targeted device, operating conditions, voltage, and other basic information. Additional pages are available from the tabs arranged at the top of the window. Except for Graph and Report, each of these pages allows you to view, edit, and add elements. The number and types of pages that are available will depend on the selected device.

Pages Available for All Device Families

Power Matrix The Power Matrix page shows the amount of power pulled by each component in the design from multiple power sources. Two tabs are provided, allowing you to view the current usage (A) and power usage (W) of each type of component. Power Calculator enables you to view the power matrix as an HTML report. It also allows you to generate a .csv file of the matrix that you can open in a spreadsheet application.

Logic Block The Logic Block page displays the estimated power consumed by the logic in the design and the factors that affect it. Power-consumption calculation in the Logic section requires both the frequency and an activity factor per clock domain. Power calculation is also based on the specified number of LUTs, distributed RAMs, ripple-carry logic circuits, and registers driven by the clock.

Clocks The Clocks page displays the estimated power consumed by the clocks in the design and the factors that affect it. Dynamic power calculation is based on the frequency of each clock.

The Clocks page only reports clocks that go to the clock tree, such as Primary Clock, Secondary Clock and Edge Clock.

For ECP5 devices, a digital clock control (DCC) column is included, which allows the operation of clock components downstream to be stopped to save power.

I/O The I/O page displays the estimated power consumed by the I/Os in the design and the factors that affect it. The power-consumption calculation for this page requires the frequency, activity factor, and number of inputs or outputs per clock domain. For bidirectional signals, it requires the number of bidirectional I/Os, input and output frequency, and input and output activity factor per clock domain. If you use an .ncd file, Power Calculator will extract the information for the I/O page directly from the .ncd file. If you do not use an .ncd file, Power Calculator will assign the default resource usage values according to the design's family.

For low-power devices, the I/O page includes the following additional columns for controlling power options:

- Power Guard (PG) – Power Guard can be turned on to enable the blocking of signals at the input buffers. Power Guard's biggest impact is in standby mode when it can be used to switch off clock inputs that are distributed using general routing resources. After you turn Power Guard on, you can use the Power Controller to enable it.
- Allow InRD Shut-off – Selecting **Yes** in this column in the Bank Voltage section allows the disabling of the dynamic referenced and differential input buffers.
- Allow LVDSO Shut-off – Selecting **Yes** in this column in the Bank Voltage section allows the LVDS output buffer driver to be turned off.

See "Controlling Power Options for Low-Power Devices" on page 790 for more information.

I/O Termination The I/O Termination page enables you to provide information about external terminations for the I/Os. You can specify the average equivalent thevenin resistive load in ohms (Rth) and the equivalent thevenin voltage in volts (Vth). Power-consumption calculation is based on the power consumed by the external termination that you provide. In certain cases, such as in a LatticeSC device, the termination can also be internal to the device.

Graph The Graph page displays three types of graphs: Power vs. VCC Supply Voltage, Power vs. Ambient Temperature, and Power vs. Frequency. Each graph displays two plots—typical and worst case. Use the Edit > Graph Settings command to change the graphs to be displayed. The dialog box enables you to specify the range, step, X axis, and Y axis of each graph. See "Generating Power Graphs" on page 795 for more information.

To prevent any unnecessary calculations and additional processing time, graphs are only generated when you select the Graph tab. During the graph generation time, you cannot change tabs and must wait for the graph calculations to finish before performing any other action. If you switch tabs and change any information that will alter the power, the graphs will be regenerated when you next select the Graph tab.

Report The Report page contains a summary of the estimated power-consumption or current-consumption data calculated by Power Calculator. Information is taken from the Power Summary and from each page of the Power Calculator user interface.

In the Power Model section, the Status can be Preliminary, Advanced, or Final.

- Preliminary – The power file has all the constants, functions, formulas, and defaults. The constants are based on nominal silicon characterization data rather than simulation data.
- Advanced – The power file has all the constants functions, formulas, and defaults. The constants are based on silicon simulation data without extracted layout parameters.
- Final/Production – The power file has all the constants, functions, formulas, and defaults. The constants are based on silicon characterization data that includes characterization of corner lots and conditions rather than simulation data.

The report is available in text (ASCII) format and in HTML format. It is updated each time you make a change to any of the data in the editable cells.

Pages Dependent on Selected Device

Block RAM The Block RAM page displays the power consumed by the embedded block RAM (EBR) in the design and the factors that affect it. Power-consumption calculation for the Block RAM page requires the frequency and activity factor per clock domain. This page separates different EBR modes into individual sections, such as single-port RAM (SP RAM), pseudo-dual-port RAM (DP RAM), true dual-port RAM (DP RAM True) and single- and dual-clock FIFOs (FIFO DC).

The Block RAM page is available for all devices except MachXO 256 and 640 and Platform Manager.

Power Modes/Avg. Power The Power Modes/Avg. Power page allows you to calculate power consumption in your device for given ratios of time spent in Normal mode and Sleep mode. To use this feature, enter a value from 0 to 100 in the **Normal Mode Time(%)** or **Sleep Mode Time(%)** field. When focus leaves that field, Power Calculator will automatically populate the other field with the remaining percentage, calculate the average power consumption, and display the result.

The Power Modes/Avg. Power page is available for LIFMD/F devices.

MIPIDPHY The MIPIDPHY page allows you to select the MIPI DPHY channel by ID (0 or 1), and then select Tx or Rx mode for that channel. Power is directly proportional to that channel's MIPI DPHY Data Rate.

EFB The EFB page enables you to specify settings for an embedded function block (EFB). The EFB block, which is connected to the device from a WISHBONE bus, includes the following peripherals: a SPI, two I2Cs, and a timer/counter (TC). For MachXO2 devices, it also includes a user flash memory (UFM). The EFB page is divided into separate sections for each of the peripherals. Frequency, activity factor, and one peripheral are required for power-consumption calculation in each section. Use of the WISHBONE bus for Timer/Counter is optional.

The EFB page is available for MachXO2, MachXO3D, MachXO3L, and Platform Manager 2 devices.

Misc The Misc page enables you to allow standby for the master clock (MCLK), which will enable you to turn off the on-chip oscillator during low-power operations, and set the frequency for MCLK. For MachXO2, MachXO3D, and MachXO3L devices, standby options for power on reset (POR) and BANDGAP are also included. Allowing standby on the Misc page enables you to use the Power Option Controller to place selected components in standby mode. See "Controlling Power Options for Low-Power Devices" on page 790 for more information. The Misc page also enables you to select and enable the mode for soft error detect (SED).

For ECP5 devices, the Misc page enables you to set the frequency and activity factor for digital temperature readout (DTR). The DTR circuit provides the junction temperature of the die while running.

For LIFMD/F devices, the Misc page enables you to place its High Frequency Oscillator (OSCI) output in Standby mode.

The Misc page is available for MachXO2, MachXO3D, MachXO3L, Platform Manager2, LIFMD/F, and ECP5.

ASC The ASC page displays the calculated power consumed by Analog Sense and Control (ASC) devices including the internal ASC component that is part of the Power Manager 2 device. The power consumed by an ASC device is determined by the number of HVOUTs used in the design. The data file generated by the Platform Designer tool includes the number of HVOUTs

used and the power model equation. The data file lists all ASCs and the power used by each and is located in the design implementation directory. This file is read by Power Calculator and then the additional power from the ASC devices is added to the total power displayed. This data is displayed in the Power Summary tab, Power by Block (W) section and the ASC page.

This ASC page is available for MachXO2, ECP5, and Platform Manager 2 devices.

Note

This data is only used when Power Calculator is in calculation mode integrated in Diamond software. If standalone Power Calculator or Power Estimator project files (.pcf) contain ASC information, it reads the project file and calculates ASC power.

PLL The PLL page displays the estimated power consumed by the phase locked loops in the design and the factors that influence it. The power-consumption calculation is based on the input frequency, the number of PLLs in the design, and the following device-dependent parameters:

- N – the feedback divider count in voltage-controlled oscillator (VCO) frequency calculation.
- V – the VCO output divider in the VCO frequency calculation
- M – the reference clock divider count in the VCO frequency calculation.

PLL standby can be enabled on the PLL page for MachXO2, MachXO3D, MachXO3L, Platform Manager2, and ECP5. This allows you to use the Power Option Controller to place selected blocks in standby mode. See “Controlling Power Options for Low-Power Devices” on page 790 for more information.

The PLL page is available for all devices except MachXO 256 and 640 and Platform Manager.

DLL The DLL page displays the estimated power consumed by the delay locked loops in the design and the factors that influence it. The power-consumption calculation for this page requires the frequency and the number of DLLs.

The DLL page is available for the following device families: LatticeECP2/M, LatticeECP3, LatticeSC/M.

DQSDLL The DQSDLL page displays the estimated power consumed by the data strobe signal delay block in the design and the factors that influence it. The power-consumption calculation for this page requires the frequency and the number of DQSDLLs driven by each clock.

The DQSDLL page is available for all device families except LatticeSC/M, MachXO, MachXO3D, ECP5, and Platform Manager.

DQS The DQS page displays the estimated power consumed by the data strobe signal block in the design and the factors that influence it. The power-consumption calculation for this page requires the frequency and the number of DQSs driven by each clock.

The DQS page is available for Platform Manager 2 and for most MachXO2 devices, including MachXO2-640U, MachXO2-1200/U and larger devices. It is also available for ECP5.

DLLDEL The DLLDEL page displays the estimated power consumed by the clock slave delay cell (DLLDEL) and the factors that influence it. The DLLDEL is used with the DQSDLL to create a 90° clock shift/delay for aligned receiver interfaces. The power-consumption calculation for this page requires the frequency and the number of DLLDELs driven by each clock.

The DLLDEL page is available for MachXO2, MachXO3D, MachXO3L, Platform Manager2, LIFMD/F, and ECP5.

DDRDLL The DDRDLL page displays the estimated power consumed by the double data rate delay locked loops in the design and the factors that influence it. The power-consumption calculation for this page requires the frequency and the DDRDLL ID.

The DDRDLL page is available for ECP5 and LIFMD/F devices.

DSP The DSP page displays the estimated power consumed by the digital signal processors in the design and the factors that affect it. The power-consumption calculation for this page requires the frequency, activity factor, and the type and number of DSPs driven by each clock.

The DSP page is available for the following device families: LatticeECP, LatticeECP2/M, LatticeECP3, ECP5, LatticeXP2.

MACO The MACO page displays the estimated power consumed by the Masked Array for Cost Optimization (MACO) blocks in the design and the factors that affect it. The power-consumption calculation for this page requires the frequency, activity factor, and the type and number of MACOs driven by each clock. For LatticeSCM devices, the type of MACO can be one of the following:

- MTCL – a memory controller core
- SPI4 – a system packet interface level 4 core
- FlexiMAC – a media access controller core
- LTSSM – a link-training and status state machine core

The MACO page is available for LatticeSCM devices.

SERDES The SERDES page displays the estimated power consumed by the serializer/deserializer SERDES blocks in the design and the factors that affect it.

The SERDES page is available for LatticeECP2M, LatticeECP2MS, LatticeECP3, ECP5UM, LatticeSC, and LatticeSCM devices. The SERDES power-consumption calculation for each of these device families requires the frequency and number of channels active in the PCS instance.

The following parameters for each PCS SERDES channel are device-dependent:

- Gearing Ratio – specifies whether the FPGA interface to the PCS SERDES block is geared to produce a 16-bit data bus or an 8-bit data bus. The Gearing Ratio column is available only for LatticeSC and LatticeSCM devices.
- TX Pre-emphasis – specifies the Tx buffer pre-emphasis level. The Tx Pre-emphasis column is available only for LatticeECP3, LatticeSC and LatticeSCM devices. See the PCS SERDES data sheet for more information on the settings.
- Mode – specifies the mode in which the PCS SERDES block operates. The Mode column is available for LatticeECP2M, LatticeECP2MS, LatticeECP3, and ECP5UM.
- Dual Channel – specifies the PCS channel for the dual-channel-based ECP5UM device.
- Receive Max Data Rate (Gbps) – specifies the maximum receive data rate. The Receive Max Data Rate column is available for ECP5UM devices.
- Rx Rate – specifies the receive rate divider: Full, Div2, or Div11. The Rx Rate column is available for ECP5UM devices.
- Tx Max Data Rate (Gbps) – specifies the maximum transmit data rate. The Tx Max Data Rate column is available for ECP5UM devices.
- Tx Rate – specifies the transmit rate divider: Full, Div2, or Div11. The Tx Rate column is available for ECP5UM devices.
- Differential Amplitude – specifies the voltage differential amplitude. The Differential Amplitude column is available for ECP5UM devices.
- Output Termination – specifies the amount of resistance for the termination. The Output Termination column is available for ECP5UM devices.
- De-emphasis Pre-cursor Select – specifies a pre-cursor differential output voltage ratio. The De-emphasis Pre-cursor Select column is available for ECP5UM devices.
- De-emphasis Post-cursor Select – specifies a post-cursor differential output voltage ratio. The De-emphasis Post-cursor Select column is available for LatticeEC4UM devices.
- Receive Loss-of-Signal Port – enables or disables the receive loss-of-signal port. The Receive Loss-of-Signal Port column is available for ECP5UM devices.
- SERDES AUX – turns on or off the TX PLL for each AUX channel of the dual-channel-base PCS SERDES. The SERDES AUX section is available for ECP5UM devices.

Color Coding of Cells

The background colors of the cells on Power Calculator pages have the following significance:

- White – The cell is editable. When you edit the contents of this type of cell, the software mode does not change. If the software is in calculation mode, it will remain in calculation mode. If the software is in estimation mode, it will remain in estimation mode.
- Gold – The cell is read-only or contains output from the software.
- Cyan (turquoise) – The cell contains data extracted from a design file, such as an .ncd file. If you enter data into this type of cell and the software is in calculation mode, it will change to estimation mode, unless you enter data into the Performance Grade, Operating Conditions, and Process Type boxes. If the software is in estimation mode, it will remain in estimation mode.
- Red – The calculated value in the Junction Temperature box is either larger than 125 degrees Celsius or larger than the maximum junction temperature allowed for a particular device and operating conditions. The Junction Temperature box is the only cell that can display a red background.

The font colors in the cells on Power Calculator pages have the following significance:

- Blue – Indicates default values.
- Red – Indicates values that cannot be edited on the I/O page. Since the I/O page has columns for inputs, outputs, and bidirectionals, red font prevents you from altering an I/O that is not valid. For example, if the I/O type belongs only to an I/O input, the cell in the # of Inputs column would display a value in black font, indicating that it is editable, but the cells in the # of Outputs and the # of Bidi columns would display values in red font to indicate that they cannot be edited.
- Black – Indicates all other text.

Working with Power Calculator Files

When you first open Power Calculator for a design project, it creates a temporary file that appears in the title bar as “Untitled.” This file contains default information, based on the device, or information extracted from the .ncd file. When you enter data into Power Calculator pages and save the changes, the information gets stored in a Power Calculator file (.pcf). You can create a new Power Calculator file (.pcf) by saving the “Untitled” file. Or you can create a new .pcf file from the File menu before or after Power Calculator is opened.

When you use a .pcf file, rather than the “Untitled” temporary file, Power Calculator maintains the information it has already extracted from the .ncd file. The saved .pcf file will not get overwritten when you rerun Map or Place & Route Design, and the software mode will not change.

This section describes how to create new Power Calculator files, how to work with multiple existing .pcf files, and how to activate a .pcf file and load it into Power Calculator. It also shows how to import a value change dump file (.vcd) and a trace report file (.twr) for power calculation.

Creating a New Power Calculator File in Diamond

You can create a new Power Calculator file (.pcf) from the File menu or from the Analysis Files folder pop-up menu. You can do this before or after opening Power Calculator.

To create a new .pcf file:

1. In Diamond, do one of the following to open the New file dialog box:
   - choose **File > New > File**
   - Press **Ctrl+N**.
   - Right-click the Analysis Files folder in the File List pane and choose **Add > New File** from the pop-up menu.
2. In the New file dialog box, select **Power Calculator Files** from the Source Files list.
3. Type a name for the new .pcf file in the name box.
4. In the Location box, enter the path and name of the directory where the Power Calculator file (.pcf) will be stored. You can use the Browse button to navigate to the desired directory. By default, the file is stored in the current project folder.

   The “Add to project” option, which will add the new .pcf file to the Analysis files folder, is selected by default. If you do not wish the .pcf file to be added to the Analysis files folder for the current project, clear the “Add to project” option. You cannot clear this option if you are using the Analysis Files folder to add a new file.
5. If your project contains more than one design implementation, select the desired implementation from the drop-down menu. By default, the active implementation is already selected.
6. Click **New** to create the .pcf file in the selected directory.

   If Power Calculator is already open and contains unsaved changes to the current .pcf file, the Confirm dialog box will appear. Click **Yes** if you want to save the changes or **No** to discard them.

   Power Calculator opens, if it is not open already. This might take a few seconds. The new .pcf file is loaded into Power Calculator and its name is displayed in the title bar. The .pcf file is added to the Analysis Files folder where its name is highlighted in bold type, indicating that it is set as the active .pcf file for the current implementation. Because the .pcf file is set as active, it will be loaded automatically when you close and reopen

Power Calculator. When you rerun a process in Diamond, the .pcf file will not get overwritten.

Note

Each time you create a new Power Calculator file through the File menu or the Analysis Files pop-up menu, the new .pcf file is automatically set as the active one for the current implementation unless you have cleared the “Add to project” option. To make the .pcf file inactive, right-click the file name in the Analysis Files folder and choose **Set as Inactive**.

See Also
- “Inputs” on page 760
- “Saving a Power Calculator File” on page 773
- “Outputs” on page 760
- “Opening an Existing Power Calculator File in Diamond” on page 774

Creating a New Power Calculator File in the Stand-Alone Power Calculator

If you have started Power Calculator as a stand-alone tool, you can create a new Power Calculator project file (.pcf) from the File menu.

To create a new .pcf file from the stand-alone Power Calculator:

1. From the stand-alone Power Calculator, choose **File > New File**.
2. In the Power Calculator – New Project dialog box, type a name for the file and browse to the desired directory.
3. To use an existing .ncd file for the project, browse to the location of the .ncd file. Otherwise, leave the .ncd box empty.
4. Click **OK**.

   If you selected an .ncd file in Step 3, Power Calculator loads the new project in calculation mode. If you did not select an .ncd file, Power Calculator loads the new project in estimation mode.

See Also
- “Inputs” on page 760
- “Saving a Power Calculator File” on page 773
- Figure on page 760
- “Opening an Existing Power Calculator File in the Stand-Alone Power Calculator” on page 775

Saving a Power Calculator File

When you enter data into Power Calculator, an asterisk appears in both the title bar and the Power Calculator tab to indicate that there are unsaved changes. If you have not yet created a Power Calculator file (.pcf) for your

project, or if no .pcf file has been set as the active file, “Untitled” will appear in the title bar. The “Untitled” file is a temporary file that contains default settings based on the device. To add power analysis changes to your project, you must save the “Untitled” as a .pcf file, open an existing .pcf file, or create a new one. Afterwards, you can simply click the Save button to write any changes to the .pcf file. You can also save an existing .pcf file to a different directory or file name.

To save the Untitled file:

1. Choose **File > Save Untitled As**, and in the dialog box, navigate to the desired directory.
2. Type a name in the file name box and click **Save**.

   The information entered in your project is saved in the .pcf file, and the .pcf file is automatically added to the Analysis Files folder in the File List pane.

To save changes to an existing Power Calculator file:

- Choose **File > Save** or press **Ctrl+S** or click  .

To save a Power Calculator file to a different directory or file name:

1. Choose **File > Save <file_name>.pcf As** to open the Save dialog box.
2. In the Save In box, navigate to the directory in which to save the .pcf file.
3. Type a different name in the File Name box.
4. In the Files of Type box, select **Power Calculator File (.pcf)**.
5. Click **Save**.

   The saved .pcf file is added to the project’s Analysis Files folder in the File List pane.

See Also
- “Adding Power Calculator Files to the Analysis Files Folder” on page 776
- “Setting a Power Calculator File as the Active Analysis File” on page 777
- “Outputs” on page 760

Opening an Existing Power Calculator File in Diamond

You can open an existing Power Calculator (.pcf) file in Diamond before or after opening Power Calculator. When you open a .pcf file in Diamond, the design information must match the information in the current project. You can open an existing .pcf file from the File menu, or from the Analysis Files folder.

To open an existing .pcf file from the File menu:

1. Choose **File > Open > File** or press **Ctrl+O**.

2. In the Open File dialog box, select **Power Calculator Files (.pcf)** from the Files of Type drop-down menu. If Power Calculator is already open, this will already be selected.
3. Navigate to the directory that contains the desired .pcf file, select the file, and click **Open**.

   Power Calculator opens automatically, if it is not already open. The Power Summary page is displayed and the information from the selected .pcf file is loaded. The name of the .pcf file appears in the title bar. If Power Calculator is in calculation mode, it will import any Bank VCCIO settings that are in the preference file and display these settings on the I/O page.

To open a recently opened .pcf file:

- Choose **File > Recent Files > *filename*** from the list of the four most recently opened files.

Note

When you open a .pcf file from the File menu, it is not automatically added to the Analysis Files folder. You must add it to the folder manually.

To open an existing .pcf file from the Analysis Files directory:

1. In Diamond, select the File List tab in the pane on the left.
2. Expand the Analysis Files folder.

   You can open the active .pcf file, if it is not already open, or one that is not active. An active .pcf file appears in bold type.
3. Double-click the name of the .pcf file that you want to open. Optionally, right-click the file name and choose **Open**.

   The information from the .pcf file is loaded into Power Calculator, and the name of the file appears in the title bar.

See Also
- "Adding Power Calculator Files to the Analysis Files Folder" on page 776
- "Setting a Power Calculator File as the Active Analysis File" on page 777

Opening an Existing Power Calculator File in the Stand-Alone Power Calculator

If you have opened Power Calculator as a stand-alone tool, you can open an existing Power Calculator file (.pcf) from the File menu.

To open an existing .pcf file from the stand-alone Power Calculator:

1. Choose **File > Open File**.

2. Navigate to the desired .pcf file and click **Open**.

   If you have unsaved changes in the currently loaded .pcf file, a Confirm message box will appear. Click **Yes** if you want to save the changes. Otherwise, click **No**.

   If an .ncd file is associated with the selected .pcf file, Power Calculator loads the project in calculation mode. If no .ncd file is associated with the .pcf file, Power Calculator loads the new project in estimation mode.

See Also ▶"Saving a Power Calculator File" on page 773

- "Creating a New Power Calculator File in the Stand-Alone Power Calculator" on page 773

Adding Power Calculator Files to the Analysis Files Folder

Diamond enables you to add existing Power Calculator files (.pcf) to the Analysis Files folder. This gives you easy access to the .pcf files in your project and enables you to designate a .pcf file as the active one for a design implementation. When a .pcf file has been set as the active analysis file, it will be loaded into Power Calculator automatically when you close Power Calculator and reopen it.

You can add a .pcf file to the Analysis Files folder before or after starting Power Calculator.

To add an existing .pcf file to the Analysis Files directory:

1. In Diamond, select the File List tab in the pane on the left.
2. Right-click the Analysis Files folder and choose **Add > Existing File** from the pop-up menu.
3. In the dialog box, select **Analysis Files (*.pcf)** in the Files of Type box.
4. Navigate to the directory that contains the desired .pcf file, select it, and click **Add**.

   The file is added to the Analysis Files folder. You can double-click the .pcf file to open it in Power Calculator.

Note

When you add a .pcf file to the Analysis Files folder, it is not automatically set as the Active .pcf file for the implementation. You must set it as the active file manually.

See Also ▶"Removing Power Calculator Files from the Analysis Files Folder" on page 777

- "Creating a New Power Calculator File in Diamond" on page 772
- "Setting a Power Calculator File as the Active Analysis File" on page 777

Removing Power Calculator Files from the Analysis Files Folder

At times, you may want to remove files from the Analysis Files folder to make room for others. When you remove a Power Calculator File (.pcf) from the Analysis Files folder, it is not deleted from your project. You can always add it back later.

To remove a .pcf file from the Analysis Files folder:

- Right-click the file name and choose **Remove**.

  The file is removed from the Analysis Files list, but it remains in your project directory.

Setting a Power Calculator File as the Active Analysis File

In Diamond, if you want Power Calculator to load a specific Power Calculator File (.pcf) each time it opens, you must designate the .pcf file as the active one for the implementation. This is done automatically when you create a new .pcf file. If no .pcf file has been set as the active one, Power Calculator will extract information from the placed and routed .ncd file when you open Power Calculator. If no .ncd file is available, it will display default resource information based on the targeted device.

To set a .pcf file as the active analysis file:

1. Make sure that you have added the desired .pcf file to the Analysis Files folder.
2. Right-click the desired .pcf file in the Analysis Files folder and choose **Set as Active Analysis File**.

   Diamond displays the .pcf file name in bold type, indicating that it is the active power analysis file for the current design implementation. Diamond does not automatically load the newly activated file into Power Calculator if a different .pcf file was already open.
3. To open the active .pcf file, if a previous file was already open, double-click the activated .pcf file name in the Analysis Files folder; or right-click it and choose **Open**.

   When you close and reopen Power Calculator for the current design project, the active .pcf file will be loaded.

Changing an Active Power Calculator File to Inactive

In Diamond, it is not required that a Power Calculator File (.pcf) be set as the active one for a design implementation. You can always open a .pcf file manually from the File menu or from the Analysis Files folder.

To change an active Power Calculator File to inactive:

- Right-click the name of the active .pcf file in the Analysis File folder and choose **Set as Inactive** from the pop-up menu.

  When you close and reopen Power Calculator, it displays default information or information from the routed .ncd file. “Untitled” appears in the title bar.

Entering Data

When you have a design open in Diamond, Power Calculator extracts information such as device, package, part, performance grade, and operating conditions. You can modify the device settings and the editable cells on any page. If Power Calculator is in calculation mode when you make any change other than Activity Factor, Frequency, Voltage, Dynamic Power Multiplier, Ambient Temperature, Performance Grade, Operating Condition, or Process Type, it will revert to estimation mode. You can revert to calculation mode after making many types of changes, if they have not been saved to the .pcf file.

Power Calculator allows you to enter data directly in the editable cells. It also enables you to make global changes to frequency and activity factors by changing the default setting in the Frequency Settings and Activity Factors Settings dialog boxes. You can also use a timing report file (.twr) to make global frequency changes or a simulation file to populate both Frequency and AF (activity factor) cells.

See Also

- “Software Mode” on page 762
- “Reverting to Calculation Mode” on page 782
- “Power Calculator Pages” on page 764

Editing Cells

Power Calculator includes built-in design rule checks. It automatically checks values that you enter into editable cells to ensure that they do not violate design rules. If you attempt to enter an inappropriate value in the Type, # I/P, # O/P, or # Bidi cell in the I/O page, Power Calculator will block the invalid value and display the previous value in the cell.

Power Calculator also provides tool tips that display the valid range of values for an editable cell. To ensure that the value you are entering is a valid one, hold your mouse over the cell to view the tool tip.

Most cells on Power Calculator pages are editable text cells that enable you to type a modified value. Others cells, such as the Device and Power parameters sections of the Power Summary page, contain visible drop-down menus for making a selection. Still others, such as those in the Type column on the I/O page, contain hidden drop-down menus that become visible when you double-click a cell.

To edit a cell:

- Depending on the type of cell you are editing, do one of the following:
  - Double-click the editable cell, type a new value, and then press **Enter** or click anywhere outside the cell.
  - Select a value from the visible drop-down list.
  - Double-click the cell and select a value from the drop-down list that appears.

  Power Calculator calculates the results automatically and displays them. It also updates the Report page.

See Also
- "Power Calculator Pages" on page 764
- "Color Coding of Cells" on page 770
- "Cutting and Pasting Cell Contents" on page 781
- "Copying and Pasting Cell Contents" on page 782
- "Changing Values Automatically" on page 782

Editing the I/O Type for LatticeSC/M Devices

For LatticeSC and LatticeSCM devices, Power Calculator provides an IO Type selection dialog box that enables you to specify an I/O type based on its properties, such as drive strength and VTT termination.

To specify the I/O type for a LatticeSC/M device based on its properties:

1. On the I/O page, select the cell in the Type column that you want to change. Make sure that you click the cell only once.

   The cell is highlighted and no drop-down arrow appears.

   Note

   If you double-click the cell, the drop-down menu will become available instead of the pop-up menu. If the drop-down arrow appears, click outside the cell, and then return and click the cell only once.

2. Right-click the highlighted cell and choose **IO Type Edit** from the pop-up menu.

The IO Type Selection dialog box lists all available I/O types in the frame on the left. The properties of each selected I/O type are shown in the frame on the right.

3. Scroll through the IO Type list, select a type, and examine its properties.
4. Repeat Step 3 until you have decided on the I/O type for the selected clock, based on the I/O type properties.
5. Click **OK** to place the selected I/O type in the cell on the I/O page.

Editing Pages

You can change the settings and values on any Power Calculator page and, if desired, save the results to a separate Power Calculator File (.pcf) in the Analysis Files folder.

To edit Power Calculator pages:

1. On the Power Summary page, modify any settings in the Device section as desired.

   When you select a different device, Power Calculator compares the design's requirements against the available resources in the selected device. If the selected device is not suitable for the design—for example, if the number of LUTs in the design exceeds those available in the device—Power Calculator will generate an error message and not allow the change. To proceed with the change, you would need to reduce the design size to fit the smaller device.
2. In the Device Power Parameters section of the Power Summary page, set the Process Type option, which specifies the process corners or conditions under which the device was manufactured. It can be one of the following:
   - Typical – specifies typical conditions to reflect the typical amount of current consumed by the circuit.
   - Worst – specifies fast conditions to reflect the maximum amount of current consumed by the circuit.
3. In the Environment section of the Power Summary page, do the following:
   - Click **Thermal Profile** to select a thermal impedance model or enter your own Effective Theta-JA value.
   - Change the ambient temperature, as desired.

   See "Controlling Operating Temperature" on page 786
4. In the Voltage/Dynamic Power Multiplier section, enter new values, as desired, for voltage and DPM.

   The voltage is the estimated power consumption by power supply. DPM is the derating factor for the dynamic portion of the power consumption.
5. Select other tabs and enter values into the editable cells of the other Power Calculator pages. The number and types of pages varies according to the device family.

6. Save your changes.

See Also ▶"Power Calculator Pages" on page 764

- Saving a Power Calculator File
- "Adding Power Calculator Files to the Analysis Files Folder" on page 776
- "Setting a Power Calculator File as the Active Analysis File" on page 777
- "Editing Cells" on page 778

Adding and Deleting Clock Rows

You can easily add and delete clock rows on the Power Calculator pages.

To add a clock row to a page:

1. Right-click inside the desired table and choose **Add Row** from the pop-up menu.
2. Enter the appropriate data in the cells that have a white or cyan (turquoise) background. Some columns with a cyan (turquoise) background, such as Type, offer drop-down menus from which you can select settings.

To delete a clock row from a page:

- Right-click in the row that you want to delete and choose **Remove Row**.

See Also ▶"Editing Cells" on page 778

Cutting and Pasting Cell Contents

You can cut the contents of a cell in a clock row and paste them in a cell in the same row or another row.

To cut and paste cell contents:

1. Double-click the desired cell to select its contents, and then right-click.
2. From the pop-up menu, select **Cut**. Alternatively, you can press **Ctrl+x**.

   You cannot cut the contents of any cells or columns that include a drop-down menu or text that is read-only.
3. Double-click the cell into which you want to paste the contents that you have cut, and then right-click.
4. From the pop-up menu, select **Paste**. Alternatively, you can press **Ctrl+v**.

See Also ▶"Editing Cells" on page 778

- "Changing Values Automatically" on page 782

Copying and Pasting Cell Contents

You can copy the contents of a cell in a clock row to a cell in the same row or another row.

To copy and paste the contents of a cell:

1. Double-click the desired cell to select its contents, and then right-click.
2. From the pop-up menu, select **Copy**. Alternatively, you can press **Ctrl+c**.
3. Double-click the cell into which you want to paste the contents that you have copied, and then right-click.
4. From the pop-up menu, select **Paste**. Alternatively, you can press **Ctrl+v**.

See Also ▶"Editing Cells" on page 778

- "Changing Values Automatically" on page 782

Changing Values Automatically

When you change values on any of the pages, Power Calculator recalculates the results automatically. For example, when you change the frequency of a clock in one cell and press Enter, Power Calculator automatically changes the frequency for that clock in all Frequency cells.

You can also use the Activity Factor Settings and Frequency Settings dialog boxes to make changes to all frequency and activity factor cells.

See Also ▶"Changing the Global Default Activity Factor" on page 783

- "Changing the Global Default Frequency Setting" on page 784
- "Importing a Value Change Dump (.vcd) File" on page 783

Reverting to Calculation Mode

After you have made changes that causes the software to run in estimation mode, Power Calculator allows you to revert to calculation mode, under the following circumstances:

- You are working with a saved .pcf file and not the "untitled" default .pcf file.
- The changes you made have not been saved to the .pcf file.

To revert to calculation mode from estimation mode:

1. Choose **Edit > Revert to Calculation Mode**.
2. In the Confirm dialog box, click **Yes** to confirm that you want to discard all the changes that you made in estimation mode.

   Power Calculator removes all the changes you made and reverts to the settings in the .pcf file.

See Also ▶"Software Mode" on page 762

Changing the Global Default Activity Factor

Power Calculator automatically assigns a global default activity factor of 10 percent in the cells of the pages that display an activity factor, such as AF (%), or that use an activity factor in calculations, such as Input AF (%). These default values appear in blue font. You can use the Edit menu to globally change this default activity factor.

To globally change the default activity factor:

1. Choose **Edit > Activity Factor Settings**.
2. In the Power Calculator - Activity Factor Settings dialog box, enter the new activity factor in the **Activity Factor Default** text box.
3. Click **OK**.

All the default activity factors appearing in blue font are changed to the new activity factor. Power Calculator automatically saves the new default.

If you manually change an activity factor in only one cell, the font becomes black to indicate that it is not a default value.

You can use a .vcd file to populate the cells that display or use activity factors. The resulting values are not considered defaults and therefore appear in black font.

See Also ▶"Activity Factor Calculation" on page 761

▶ "Importing a Value Change Dump (.vcd) File" on page 783

Importing a Value Change Dump (.vcd) File

Power Calculator enables you to import a value change dump (.vcd) file of simulation results into your project. Normally, you would import a .vcd file only when you want the Frequency and AF (activity factor) cells on Power Calculator pages to be populated with frequency and activity factor data from the .vcd file.

To ensure that Power Calculator populates the Frequency and AF cells with the VCD information, make sure that you follow these requirements:

- The .vcd file should be in the format of gate-level simulation, and it should match the design.
- A post-PAR timing simulation netlist must be used for name matching. You cannot use the RTL design.
- A stimulus must be used in the simulator that actually toggles the signals you are interested in; otherwise, you will see no difference in Power Calculator after the VCD is read.

To import a .vcd file into your project:

1. Choose **Edit > Open Simulation File** to open the Power Calculator – Open Simulation File dialog box.
2. In the VCD File box, type or select the path and name of the .vcd file that you want to open.
3. In the Module Name in VCD box, specify the name of the module in the .vcd file from which to take the frequency and activity factor data.
4. Select the Case Sensitive option if the name of the .vcd file to be imported is case-sensitive.
5. Click **OK**.

   Power Calculator's Frequency and AF cells are now populated with the data from the .vcd file.

See Also ▶"Changing the Global Default Frequency Setting" on page 784

Changing the Global Default Frequency Setting

You can globally change the default frequency values for the clocks listed in the Power Calculator pages. These default values appear in blue font in the Freq. (MHz) columns.

You can change this global frequency value either by specifying a value or by indicating that the frequency value be taken from the constrained or actual frequency in the trace report (.twr) file.

To globally change the default frequency setting by specifying a value:

1. Choose **Edit** > **Frequency Settings**.
2. In the Power Calculator - Frequency Settings dialog box, enter the new activity factor in the Frequency Default cell, in megahertz.

   The default is 0 megahertz.
3. Click **OK**.

To globally change the default frequency setting by using values from the .twr file:

1. Choose **Edit** > **Frequency Settings**.
2. In the Power Calculator - Frequency Settings dialog box, select one of the following options in the Frequency TWR box. For all these options, the .twr file must contain the names of the clocks in the pages for which you want default frequency values.
   - Minimum of Preference And Trace – Specifies that the default frequency in the Frequency (MHz) column of the Power Calculator pages be taken from the lesser of the constrained frequency or the actual frequency in the trace report (.twr) file, if you have imported this file into your project. The frequency is in megahertz.

- Always Use Preference – Specifies that the default frequency in the Frequency (MHz) column of the Power Calculator pages be taken from the constrained frequency in the trace report (.twr) file, if you have imported this file into your project. The frequency is in megahertz.
- Always Use Trace – Specifies that the default frequency in the Frequency (MHz) column of the Power Calculator pages be taken from the actual frequency in the trace report (.twr) file, if you have imported this file into your project. The frequency is in megahertz.

3. Click **OK**.

   Power Calculator now populates the Frequency cells of its pages with the frequency data from the imported .twr file.

See Also ▶"Inputs" on page 760

Estimating Resource Usage

Power Calculator allows you to specify an estimate of resources that the design will use, for the purpose of power analysis. The estimate can be based on design type or on component utilization. Based on your selections, Power Calculator immediately displays the number of resources that will be utilized for each type.

To estimate resource usage based on design type:

1. Choose **Edit > Resource Settings**.
2. In the dialog box, select **Specify Resource by Design Type**.
3. Select the design type from the drop-down menu.

   Power Calculator calculates the resource usage based on the design and displays the utilization in the bottom portion of the dialog box.
4. Click **OK**.

To estimate resource usage based on component utilization:

1. Choose **Edit > Resource Settings**.
2. In the dialog box, select **Specify Resource by Component Utilization**.
3. Do one or both of the following:
   - Select a Small, Medium, or Large option based on the design size.

     Power Calculator displays a percentage of Logic, I/O, and EBR based on your selection.
   - Select a percentage from the Logic, I/O, and EBR drop-down menus.

   Power Calculator calculates the resource usage based on your selections and displays the utilization in the bottom portion of the dialog box.
4. Click **OK**.

See Also ▶"Estimating Routing Resource Usage" on page 786

Estimating Routing Resource Usage

You can estimate the amount of routing resources that your design will use, for the purpose of power analysis.

To estimate routing resource usage:

1. Choose **Edit > Estimation Mode Settings**.

   The Power Calculator - Estimation Mode Settings dialog box appears.

2. From the Routing Resource Utilization drop-down menu, select the amount of routing resources that you expect your design to use. You can select from the following:

   - Low – Uses a small amount of routing resources.
   - Medium – Uses an average amount of routing resources. This setting is the default.
   - High – Uses a large amount of routing resources.

3. Click **OK**.

See Also ▶"Estimating Resource Usage" on page 785

Controlling Operating Temperature

Minimizing the device's operating temperature is critical to reducing power consumption.

A device has two parts: the die, which is the silicon inside the device, and the package, which is the outer shell. Each die-package combination has a thermal resistance value (often referred to as theta), which is a measure of how well the combination can dissipate heat. Lower values indicate better heat dissipation for the device.

Thermal impedance is the cumulative individual thermal resistances of a defined network. Power Calculator offers different models that provide ways to calculate the thermal impedance for a given device when it is mounted on the board. These models cover scenarios related to board sizes, air flows, and heat sinks that affect the thermal impedance. You can use these models to calculate the thermal impedance for the scenario that you choose.

You can choose the thermal impedance models by clicking the **Thermal Profile** button in the Environment section at the top right of the Power Summary page.

These thermal impedance models use the following terminology:

- Junction temperature – the temperature of the die in the device package, in degrees Celsius. You can adjust the junction temperature by choosing a

model that applies a heat sink and changes the air flow value. Junction temperature is also affected by the package that you select in the Package Type box. In addition, the changes that you enter in many of the editable (white and turquoise) cells on the Power Summary page affect junction temperature.

- Heat sink – any material or object that dissipates unwanted heat from a device by absorbing it and conducting it away to a surface from which it dissipates into its surroundings. The reduction of junction-to-ambient thermal impedance depends on different factors, such as the speed and direction of the air flow over the heat sink and the materials used to attach the heat sink to the package. For heat-sink properties and proper attachment methods, contact your heat-sink manufacturer for specifications.
- Air flow – the movement of air around the device in a package to cool it. It is measured in linear feet per minute (LFM). The higher the air flow value you select, the greater the cooling effect on the device.
- Ambient temperature – the expected operating temperature, in degrees Celsius, of the medium surrounding a device in a package.
- Theta JA – the thermal impedance between the silicon die and the ambient air within a JEDEC-defined environment. The boards used to measure these values have four layers, and their size is defined by JEDEC specifications.
- Effective Theta JA – similar to Theta JA, but defined as the sum of all the package and board thermal resistances outside of a JEDEC-defined environment. It indicates how well the heat dissipates from the die to the ambient (air) for a particular thermal network as a whole outside of a JEDEC-defined environment.
- Theta JB – indicates how well the heat dissipates from the junction on the silicon die to the board.
- Theta JC – indicates how well the heat dissipates from the junction of the die to the package case in which it is enclosed, as defined by the JEDEC specifications.
- Theta BA – indicates how well the heat dissipates from the board to the ambient (air).
- Theta CS – indicates how well the heat dissipates from the package case to the heat sink. It is a measure of the thermal resistance of the interface material that makes contact between the package case and the heat sink attached to the package. It can be thermal grease, double-sided sticky tape, glue, or phase-shift material.
- Theta SA – indicates how well the heat dissipates from the heat sink to the ambient (air).

See Also ▶"Power Calculator Pages" on page 764

Selecting a Thermal Impedance Model

You can experiment with various board sizes, heat sinks, and air flow settings to select a thermal impedance model for your design. The choice of board affects the Theta JB and Theta BA values, and the heat sink and air-flow selections affect the Theta SA and Theta JA values.

Note

The current values provided with the thermal models are averages of different values, and they are provided as a courtesy. To produce better predictions, it is advised that you submit your own thermal values.

To select a thermal impedance model:

1. In the Environment section of the Power Summary page, click **Thermal Profile**.

   The Effective Theta-JA specifies the cumulative thermal impedance of a particular system. This figure is used in calculating the junction temperature (Tj) of a die in a particular environment according to the following formula:

   Tj = *power* * *theta_effective* + *ambient_temperature*.

2. In the Power Calculator - Thermal Profile dialog box, select the source of the effective thermal impedance (effective Theta JA) value:

   - If you want Power Calculator to calculate the effective Theta JA value from the selections that you make in the Board Selection, Heat Sink Selection, and Airflow Selection drop-down menus, choose **Use Thermal Models**.
   - If you want to provide your own effective Theta JA value, choose **User-Defined Theta-JA Effective**.

3. If you chose Use Thermal Models:

   a. Select the size of the board that you will be using from the Board Selection drop-down menu:

      - JEDEC Board (2S2P) – Specifies that a board defined by JEDEC specifications be used. These specifications are based on real boards and measurements conducted in the lab used by Lattice Semiconductor. For packages < 27.0 mm in length, the buried planes are 74.2 mm x 74.2 mm (3” x 3”). For packages larger than or equal to 27.0 mm, the buried planes are 99.6 mm x 99.6 mm (4” by 4”). Power Calculator uses Theta JA or Theta JC, depending on the type of heat sink selection you make in the Heat Sink Selection menu. Theta JA is used only if you choose No Heat Sink. The ThetaJA value is the published value measured in the lab.
      - Small Board – Specifies that a board that is slightly larger than the JEDEC board be used. The board is assumed to be 6” to 8” square. This setting adds the Theta JB value to the board thermal impedance.

- Medium Board – Specifies that a medium board be used, one that is assumed to be 8” to 12” square. This setting adds the Theta JB value to the board thermal impedance.
- Large Board – Specifies that a large board be used, one that is assumed to be larger than 14” square. This setting adds the Theta JB value to the board thermal impedance.

Note

Many factors can affect the actual thermal properties and change the values: the number of board layers, airflow over the board, number of devices powered up around the device package and their distance from the package, the thickness of the copper and the width of the traces on each layer, the number of thermal vias, and the shape and thickness of each power and ground plane below the transfer of heat from the die to the ambient environment. Users are encouraged to make their own thermal measurements or simulations and use them in Power Calculator to see what kind of junction temperature might result.

b. Select the type of heat sink from the Heat Sink Selection drop-down menu:
   - No Heat Sink – Specifies that no heat sink be used. Theta JA is used, and you must choose the air flow from the Airflow Selection menu. The No Heat Sink setting is the default.
   - Low-Profile Heat Sink – Specifies that a short heat sink be used.
   - Medium-Profile Heat Sink – Specifies that a medium heat sink be used. This setting adds the Theta JC value to the Heat Sink thermal impedance.
   - High-Profile Heat Sink – Specifies that a tall heat sink be used.
   - Custom-Profile Heat Sink – Enables you to specify your own heat-sink value in the Theta-SA for Custom Heat Sink box. The Airflow Selection option is not available when you select Custom-Profile Heat Sink.

c. Select the air flow in linear feet per minute (LFM) from the Airflow Selection drop-down menu:
   - 0 LFM
   - 200 LFM
   - 500 LFM

   If you use a heat sink, the 0 LFM setting is not available.

   The Airflow Selection option is not available when you select Custom-Profile Heat Sink from the Heat Sink Selection menu.

4. If you chose User-Defined Theta-JA Effective, enter your effective thermal impedance value in the Effective Theta-JA box. Values entered must be greater than 0.

   This box is not available if you selected Use Thermal Models.

5. Click **OK**.

The new effective Theta JA value now appears in the Effective Theta-JA box in the Environment section of the Power Summary page.

See Also ▶"Controlling Operating Temperature" on page 786

- "Changing the Ambient Temperature" on page 790

Changing the Ambient Temperature

The ambient temperature is the expected operating temperature, in degrees Celsius, of the medium surrounding a device in a package. You can select an ambient temperature in the range of -40 to 125 degrees Celsius.

To change the ambient temperature:

- In the Environment section of the Power Summary page, enter a value, in degrees Celsius, in the Ambient Temperature box and press **Enter** or click anywhere outside the cell.

  The value entered must be between -40 and +125 degrees Celsius. The default value is 25 degrees Celsius for all devices.

When you change the value in the Ambient Temperature box, Power Calculator updates the following cells:

- Junction Temperature
- Values in the Static and Total columns of the Current by Power Supply Details and Power by Power Supply Details sections
- The totals in the Power by Block Column

If you enter a value that is beyond the commercial, industrial, or automotive device limits, you will receive an error message that displays the range of valid values.

See Also ▶"Controlling Operating Temperature" on page 786

- "Selecting a Thermal Impedance Model" on page 788

Controlling Power Options for Low-Power Devices

The MachXO2, MachXO3D, MachXO3L, ECP5, and Platform Manager 2 devices include many low-power architectural features. Power Calculator provides tools for taking advantage of these features, including the Power Option Controller. The Power Option Controller enables you to dynamically change the power of certain blocks through the use of standby mode and other options. A chart is provided that compares the amount of power used by components when standby is on and when it is off.

Three power modules are available, which you can generate from IPexpress: Power Controller, Dynamic Bank Controller, and Power Guard. To get the full benefit of Power Calculator's Power Option Controller, it is recommended that

you add these modules to your design. You can still use the Power Option Controller without adding these modules, but the calculations will be in estimation mode only.

Note

Power Guard is not available for ECP5.

For more information about these power modules, refer to the "Lattice Module Reference Guide" on page 1400.

Added Features for Low-Power Architecture

Power Calculator provides an EFB page for the low-power MachXO2, MachXO3D, MachXO3L, and Platform Manager 2 devices. The EFB page enables you to specify settings for an embedded function block (EFB). The EFB block, which is connected to the device from a WISHBONE bus, includes a SPI, two I2Cs, and a timer/counter peripheral. For higher-density devices, the EFB also includes user flash memory (UFM).

Other pages—including I/O, PLL, and Misc—allow you to set standby mode or power guard (PG) for components that you want to allow to be dynamically turned off during low-power operation modes. You can then use the "Power Option Controller" on page 791 to select the components to be turned off. Many of these features are also available for ECP5 devices.

Power is calculated with a precision of six decimal places instead of four.

See Also "Power Calculator Pages" on page 764

- TN1198, *Power Estimation and Management for MachXO2 Devices*
- TN1289, *Power Estimation and Management for MachXO3L Devices*
- TN1307, *Power Management and Calculation for CrossLink Devices*

Power Option Controller

The Power Option Controller gives you control over power management features. It enables you to conserve power by turning on and off blocks in the chip that consume power, manage the entrances and exits of signals, and use power guard (PG) to stop signals from entering the chip. First, set the standby, shut-off, and power guard options that you want to allow, using the appropriate Power Calculator pages. Afterwards, return to the Power Summary page, open the Power Option Controller, and select the elements to be turned off during low-power operation.

For MachXO2, MachXO3D, MachXO3L, and Platform Manager 2 devices, the Power Option Controller includes the following options for turning off elements that have been placed in standby mode:

- Disable Bandgap – turns off the Bandgap. When this options is selected, analog circuitry such as the PLLs, on-chip oscillator, and referenced and differential I/O buffers are also turned off.
- Disable POR – turns off the power-on-reset circuit, which monitors VCC levels. When the POR circuitry is turned off, limited power detection circuitry is still active. This option is only recommended for applications in which the power supply rails are reliable.
- Disable OSC – turns off the on-chip oscillator.
- Disable PLL – turns off the selected phase-locked loop.

The Power Option Controller includes the following dynamic bank options:

- Disable InRd – turns off the referenced and differential input buffers for a selected bank.
- Disable LVDSO – turns off the LVDS output buffer for a selected bank.
- Enable PG – enables Power Guard for the selected bank.

For ECP5 devices, the following elements that have been placed in standby can be turned off in the Power Option Controller: OSC, PLL, InRd, LVDSO, DDRDLL.

For LIFMD/F devices, the following elements that have been placed in standby can be turned off in the Power Option Controller: InRd, LVDSO, PLL, OSC, and MIPI DPHY.

See TN1198, *Power Estimation and Management for MachXO2 Devices*

TN1289, *Power Estimation and Management for MachXO3L Devices*

See Also ▶"Power Calculator Pages" on page 764

- "Lattice Module Reference Guide" on page 1400

Enabling Low-Power Options

For the low-power MachXO2, MachXO3D, MachXO3L, Platform Manager 2, and ECP5 devices, Power Calculator enables you to set options for selected components that will allow them to be dynamically turned off during low-power operation modes. The options include Allow Standby, Allow Shut-off, and Power Guard. You first allow these options for selected components by using the appropriate Power Calculator pages. Afterwards, the Power Option Controller enables you to choose those that you want to be dynamically turned off for low-power operations.

To enable the Power Option Controller low-power options:

1. Select the **Misc** tab. In the MCLK Allow Standby column, choose **YES** if you want to allow standby for the on-chip oscillator. Choose **Yes** for the other standby options that you want to allow.

2. Select the **PLL** tab. In the Allow Standby column, choose **Yes** for each of the PLL standby options that you want to allow.
3. Select the **I/O** tab and do the following:
   a. In the PG column of the Inputs and Outputs section, select the input clock signals for which you want to turn on the power guard, and then choose **On**. Do the same in the Bidirectional I/Os section.

      The power guard settings are displayed in the read-only PG column of the Bank Voltage section.

   b. For each bank in the Bank Voltage section, choose **Yes** in the Allow InRD Shut-off column if you want to allow the referenced and differential input buffers to be turned off.

   c. In the Allow LVDSO Shut-off column of the Bank Voltage section, choose **Yes** for a selected bank if you want to allow the LVDS output buffer driver to be turned off.
4. On the Power Summary page, click the **Power Control** button.

   The Power Option Controller shows the dynamic power options that you have allowed on the Power Calculator pages. Options that you have not allowed remain dimmed.
5. Select the desired dynamic power options that have been enabled and click **OK**.

See Also ▶"Power Calculator Pages" on page 764

- "Power Option Controller" on page 791
- "Lattice Module Reference Guide" on page 1400
- TN1198, *Power Estimation and Management for MachXO2 Devices*
- TN1289, *Power Estimation and Management for MachXO3L Devices*
- TN1307, *Power Management and Calculation for CrossLink Devices*

Comparing Power Results for Standby Mode

For the low-power MachXO2,MachXO3D, MachXO3L, ECP5, and Platform Manager 2 devices, Power Calculator estimates the amount of power used by components that can be placed in standby mode and compares this usage with non-standby mode. The results are presented in a comparison chart that you can access from the Edit menu.

The comparison chart gives you a good idea of the amount of power that can be saved by enabling standby mode for the different components.

To compare power results for standby mode:

- Choose **Edit > Comparison chart of power awareness**.

The Comparison chart of power (W) displays the standby and non-standby power usage for each type of component that can be placed in standby mode for the targeted device.

See Also ▶"Obtaining Time-Based Average Power Usage" on page 794

Obtaining Time-Based Average Power Usage

When you target a MachXO2, MachXO3D, MachXO3L, ECP5, or Platform Manager 2 device, you will typically be using standby mode, regular mode, and shutdown mode for various periods of time. Power Calculator enables you to obtain an estimate of the average power used, based on the percentage of time taken by each of these modes.

To obtain an estimate of time-based average power usage:

1. Choose **Edit > Average power and thermal over time**.
2. In the dialog box, type an estimated percentage of time that will be taken by shutdown mode (T1), standby mode (T2), and full power mode (T3). The percentage total for all three modes must equal 100.

   Power Calculator calculates the average power usage and displays it beneath the equation at the bottom.

See Also ▶"Comparing Power Results for Standby Mode" on page 793

Comparing Power Consumption Among Multiple Implementations

Power Calculator enables you to compare the power consumption among multiple implementations of your design. Each implementation must target the same device and have at least one power calculator project file (.pcf).

If you are using Power Calculator within the Diamond environment, Power Calculator will display the comparison table using the .pcf files that are in the Analysis Files folder for each implementation. If you are using the stand-alone Power Calculator, you will need to add the .pcf files.

To compare power consumption among multiple design implementations:

- Choose **Edit > Compare power of implementations**.

  Power Calculator generates a comparison table of power usage for all the implementations that contain .pcf files. If an implementation contains more than one .pcf file in the Analysis Files folder, the table will show multiple comparisons for the implementation.

- Click **View HTML** to view the table in your browser.
- To save the table as a text file, click **Generate Text Report**.

Viewing and Printing Results

In addition to the automatically generated reports from power settings, Power Calculator provides graphs of power consumption and enables you to print information from the pages and generate a comma-separate value file (.csv) from the command line.

Generating Power Graphs

Power Calculator can create graphs showing how power consumption is affected when you vary the voltage, temperature, and clock frequency in the design. You can generate three default power graphs on the Graph page: Power vs. Supply Voltage, Power vs. Ambient Temperature, and Power vs. Frequency. Each graph displays a typical and worst case plot. You can select the X axis and Y axis for each graph.

Graphs are only generated when you select the Graph tab. During graph generation, you cannot select a different tab until the calculations are finished. If you change the information in any other page that alters the power, the graphs will be regenerated when you again select the Graph tab. If you do not change any power information, the graphs will not be regenerated.

To generate the Power vs. Supply Voltage graphs:

1. Choose **Edit > Graph Settings**.
2. In the Power by Section part of the Graphs Settings dialog box, set the following options:
   a. In the Y Axis box, select Total Power or the type of block power, such as I/O or Block RAM, to place on the Y axis of the graph.
   b. In the X Axis box, select the type of supply voltage to place on the X axis of the graph.
   c. In the Lower Limit box, enter the lower boundary of the voltage range on the X axis. The lower limit can be 5 percent lower than the nominal supply value.
   d. In the Upper Limit box, enter the upper boundary of the voltage range on the X axis. The upper limit can be 5 percent higher than the nominal supply value.
   e. In the Resolution box, enter the step in which the supply voltage values on the X axis should appear. The voltage supply step is limited to a resolution of .01.
3. Click **OK** to close the dialog box and apply the settings.
4. Click the Graph tab to see the resulting charts.

To generate the Power vs. Ambient Temperature graphs:

1. Choose **Edit > Graph Settings**.
2. In the Power by Temperature part of the Graphs Settings dialog box, set the following options:
   a. In the Y Axis box, select Total Power or the type of block power, such as I/O or Block RAM, to place on the Y axis of the graph.
   b. In the X Axis box, select **Ambient Temperature** for the X axis of the graph. Temperature is in degrees Celsius.
   c. In the Lower Limit box, enter the lower boundary of the temperature range on the X axis. The range limits are determined by the device's operating condition:
      - Industrial: –40 through 105
      - Commercial: 0 through 85
      - Automotive: –40 through 125
   d. In the Upper Limit box, enter the upper boundary of the temperature range on the X axis. The range limits are determined by the device's operating condition:
      - Industrial: –40 through 105
      - Commercial: 0 through 85
      - Automotive: –40 through 125
   e. In the Resolution box, enter the step in which the temperature values on the X axis should appear. The temperature step is limited to resolution of 10 degrees Celsius.
3. Click **OK** to close the dialog box and apply the settings.
4. Click the Graph tab to see the resulting charts.

To generate the Power vs. Frequency graphs:

1. Choose **Edit > Graph Settings**.
2. In the Power by Frequency part of the Graphs Settings dialog box, set the following options:
   a. In the Y Axis box, select Total Power or the type of block power, such as I/O or Block RAM, to place on the Y axis of the graph.
   b. In the X Axis box, select the clock to place on the X axis of the graph. It can be any of the clocks listed on the Clocks page.
   c. In the Lower Limit box, enter the lower boundary of the frequency range on the X axis. The lower boundary is limited to 0 MHz.
   d. In the Upper Limit box, enter the upper boundary of the frequency range on the X axis. The upper boundary is limited to 10000 MHz.
   e. In the Resolution box, enter the step in which the frequency values on the X axis should appear. The default frequency increment is 20 MHz.
3. Click **OK** to close the dialog box and apply the settings.

4. Click the Graph tab to see the resulting charts.

Viewing the Power Calculator Report

The report page contains a summary of the estimated power-consumption or current-consumption data calculated by Power Calculator. It is available in text (ASCII) format and in HTML format, and it is updated each time you make a change to any of the data in the editable cells.

To view the Power Calculator report:

- To view the results in text format, click the Report tab.
- To view the results in HTML format, click the Report tab, and then click **View HTML Report**.

See Also "Entering Data" on page 778

"Printing Information" on page 797

Printing Information

After calculating power, you can print the results displayed on any of the pages, including the Report and Graph pages. Optionally, you can preview pages before printing.

To preview and print:

1. In the Power Calculator window, click the tab of the desired page.
2. Choose **File > Print Preview** to activate the Print Preview dialog box.

   Use the zoom tools and the Fit Page and Fit Width buttons on the toolbar to position the page in the window.

   When there are multiple pages, use the Next, Last, Previous, and First buttons to navigate through them.
3. Select the Portrait or Landscape button on the toolbar.
4. Click **Print** to queue the results to a printer.
5. In the Print dialog box, click **Print**.

To print information from a page:

1. In the Power Calculator window, click the tab of the desired page.
2. Choose **File > Print**.
3. In the dialog box, click **Print**.

See Also "Entering Data" on page 778

"Power Calculator Pages" on page 764

Chapter 11

Analyzing Signal Integrity

Signal integrity analysis enables characterization of interconnect discontinuities and behaviors. You can use signal integrity analysis models to perform a simulation on your board or to trace setup before laying out the board to look for signal integrity issues, such as crosstalk, reflection, ringing, overshoot, undershoot, impedance mismatch, and line termination.

Lattice Semiconductor supports signal integrity (SI) analysis at the PC board level using IBIS or HSPICE format models. For information on its PCB design support, see the following Web site:

http://www.latticesemi.com/support/pcbdesignsupport.cfm

Lattice Semiconductor IBIS Models

The Input/Output Buffer Information Specification (IBIS) is a a behavioral model used for simulation. It was developed originally by Intel in the early 1990s and has become a device-modeling standard that is regulated and developed by a forum of electronic design automation (EDA) vendors, computer manufacturers, semiconductor vendors, universities, and end users.

IBIS is a standard supported by all popular signal integrity analysis tool vendors, including Cadence, Mentor Graphics, and Zuken. You can generate design-specific IBIS models for an FPGA project from Lattice Diamond or you can download general-purpose models by family on the IBIS Models by Product web page on the Lattice web site.

About IBIS Models

IBIS models help you analyze signal integrity and electromagnetic compatibility (EMC) on printed circuit boards. Unlike conventional behavioral simulation models that contain such modeling elements as schematic symbols and polynomial expressions, IBIS models consist of tabular data. This tabular data contains current and voltage values in the output and input pins, as well as the voltage and time relationship at the output pins under rising or falling switching conditions. The IBIS model data represents realistic

device behavior, based on collected data at varied operating conditions. IBIS models provide this data without disclosure of any proprietary design information.

IBIS provides more accurate models because it accounts for non-linear aspects of the I/O structures, the ESD structures, and the package parasitics. It has several advantages over other traditional models such as SPICE, including faster simulation times and no non-convergence problems. Finally, IBIS can be run on nearly any industry-wide platform because of an almost universal EDA vendor support for the IBIS standard.

Lattice Semiconductor IBIS Model Files

In the Lattice Diamond software, Lattice Semiconductor provides IBIS model files for behavioral simulation. IBIS models are located in the following installation directory path in Diamond:

<*install_dir*>/cae_library/ibis

The following table lists all available IBIS model files in the software.

Table 6: Available Lattice IBIS Models

Family	Lattice Semiconductor IBIS Model Files
LatticeEC	ec.ibs
LatticeECP	ecp.ibs
LatticeECP2	ecp2.ibs
LatticeECP2M	ecp2m.ibs
LatticeECP2MS	ecp2ms.ibs
LatticeECP2S	ecp2s.ibs
LatticeECP3	ecp3.ibs
ECP5	ecp5u.ibs
LIFMD	lifmd.ibs
MachXO	machxo.ibs
MachXO2	machxo2.ibs
MachXO3L	machxo3l.ibs
LatticeSC	sc.ibs
LatticeXP	xp.ibs
LatticeXP2	xp2.ibs
Platform Manager	platform_manager.ibs

An IBIS model file is not an executable; it is an ASCII file of all the pertinent data that represents a device's electrical behavior and can be used in a simulator. IBIS files consist of three primary parts:

- file header information that describes the file (revision, date, conventions, source), the device, and the company
- device name, pinout, and pin-to-buffer mapping
- I/O-voltage (I/V) and voltage-time (V/T) data for each model

Lattice Semiconductor IBIS models characterize all available devices in a given device family. Files contain multiple data matrixes that contain similar data for a given device in the same family.

Note

The collected IBIS source code data in these files is only representative of how a given Lattice Semiconductor device operates. It is still your responsibility to verify your design for consistency and functionality through the use of formal verification methods. There are no guarantees regarding the use or functionality of the IBIS data in the files listed in Table 6.

Lattice Semiconductor IBIS-AMI Model Files

In the Diamond software, Lattice Semiconductor provides IBIS Algorithmic Modeling Interface (IBIS-AMI) models for SERDES, which enable fast and accurate behavioral simulation. Regular IBIS models are not suitable for modeling high speed interfaces such as SERDES, because they lack the ability to emulate its wave shaping techniques designed to counteract the detrimental effects of the channel on the transmitted signal. IBIS-AMI models resolve this issue by enhancing a normal IBIS model with an algorithmic model that provides these essential features.

Two IBIS-AMI models are provided: one that models the SERDES transmitter, and one that models the SERDES receiver. Each IBIS-AMI model contains three parts:

- a regular IBIS model of the SERDES that is used as the base of the IBIS-AMI modes
- a compiled object file (*.dll or *.so) that contains the algorithmic model, which takes in the user parameters and applies pre-emphasis on the transmitter side and equalization and gain on the receiver side
- a parameter file used by the compiled object to choose different levels of emphasis, equalization, or gain

The IBIS-AMI models are located in the following installation directory path in Diamond:

<*install_dir*>/cae_library/ibis_ami

The following table lists all available IBIS-AMI model files in the software for PC and Linux.

Table 7: Available Lattice IBIS-AMI Models

Family	Lattice Semiconductor IBIS-AMI Model Files
ECP5	ecp5um_serdes.ibs
	LSC_ecp5um_rx_ami.dll
	LSC_ecp5um_rx_ami.so
	LSC_ecp5um_tx_ami.dll
	LSC_ecp5um_tx_ami.so
	[userdesign]_rx_params.ami
	[userdesign]_tx_params.ami

Note

The following platforms and tools are supported for Lalttice Semiconductor IBIS-AMI models:

Hyperlynx on Windows 7 (both 32 bits and 64 bits)
Hyperlynx on Linux (RedHat 3 and RedHat 4 64 bits
SystemSi on Linux (RedHat 3 and RedHat 4 64 bits)

Generating IBIS Models in Lattice Diamond

You can generate your own IBIS models by double-clicking the IBIS Model process or IBIS-AMI process beneath Export Files in the Process view. This process generates a design-specific model file, *project_name*.ibs, or project_name.ami, in your project directory.

HSPICE Models for FPGA Devices

Encrypted HSPICE models provide support for Lattice Semiconductor device families that feature SERDES input and output buffers. Encrypted HSPICE provides very accurate models based on the source Lattice Semiconductor chip design and can portray dynamic behavior of buffers like SERDES.

For high-speed interfaces over 1 gigahertz, the HSPICE I/O Kit is available by request from the following Web site:

http://www.latticesemi.com/support/hspiceiokit.cfm

Chapter 12

Programming the FPGA

After you have created and verified your design, you can use the final output data file (JEDEC or bitstream) to download or upload a bitstream to or from an FPGA device using Diamond Programmer.

Diamond Programmer supports serial, concurrent (turbo), and microprocessor programming of Lattice devices in PC and Linux environments. Device chains can be scanned automatically using the Diamond Programmer graphical user interface.

Diamond Programmer is integrated into the Diamond software environment, and is also available in a stand-alone version.

Features include:

- Scan chain and display chain contents (.xcf file)
- Download data files to devices
- Create, modify, and display an .xcf file
- Generate files based on the .xcf file (Tcl or command line only)

Diamond Programmer uses a Single Document Interface (SDI) where a single .xcf project is displayed per Diamond Programmer instance. Opening additional .xcf files in Diamond-integrated mode closes the current .xcf and opens the specified .xcf. To open additional .xcf files in stand-alone mode, a separate instance of Diamond Programmer needs to be launched.

The main view displays the devices in the current Diamond Programmer project resulting from the JTAG Scan action or from manual creation in a table.

Double-clicking on an uneditable cell or right-clicking and choosing **Device Properties** opens the Device Properties Dialog Box, which displays more information about the selected device. Additionally, some entries may be edited directly on the table by clicking.

Columns can be displayed or hidden by choosing **View > Columns**. The default columns that are displayed are Process, Status, Device Family, Device, Operation, File Name, File Date/Time, Checksum, USERCODE, BSDL File Name, and Verbose Logging.

Additional columns are available, but are hidden by default. The columns are Device Vendor, Device Full Name, IR Length, Device ID, Device Package, and Device Description.

The following columns are directly editable from the table:

- Enable
- Device Family
- Device
- File Name
- Verbose Logging

Some of these columns become uneditable (grayed out) if the selected operation or other option does not support it. For example, File Name will be grayed out for a Bypass operation.

Each row has a column for the device status. The status indicates whether the operation performed was successful or not. This field is also used for read back operations to display what is read back if the data to display is short. For larger data sets that are read back, a dialog box is displayed.

See Also

- "Programming the FPGA" on page 802
- "File Formats" on page 803
- "Serial Peripheral Interface (SPI) Flash Support" on page 804
- "Programming Lattice Mature Devices with Stand-Alone Diamond Programmer" on page 816
- "Using Diamond Programmer" on page 824
- "Programmer Options" on page 856

File Formats

Lattice supports the following file formats.

Serial Vector Format File An ASCII file (with the extension .svf) that stores programming data for programming one or more fixed algorithm devices in Automated Test Equipment (ATE)-type programming environments.

Using SVF files is a way of exchanging data between the Automated Test Equipment (ATE)-type programming environment and the software driving it. This device is used to invoke the SVF processor to process the SVF file. If BYPASS is selected, other devices in the chain are processed and the SVF file is bypassed.

Current SVF specifications are available from the Asset-Intertech website at www.asset-intertech.com.

Data File A data file can be a JEDEC, ISC, hex, or bitstream file. Each of these files is based upon an IEEE programming standard:

JEDEC The programming standard from the Joint Electron Design Engineering Committee, a committee of programmer and semiconductor manufacturers that provide common standards for programmable issues. Examples include acceptable test characters for PLDs and standard data transfer/programming formats for PLDs. The JEDEC Standard is the industry standard for PLD formats. In Diamond Programmer, JEDEC usually refers to the JEDEC fuse map of your design for the device that you have selected.

ISC The In System Configuration standard, based upon the IEEE 1532 standard for programming one or more compliant devices currently while mounted on a board or embedded in a system.

Bitstream Data files used for configuring the volatile memory (SRAM) of FPGAs.

Hex Hexadecimal PROM data files used for Programming into external non-volatile memory, such as parallel or SPI Flash devices.

See Also

- “Programming the FPGA” on page 802
- “Using Diamond Programmer” on page 824

Serial Peripheral Interface (SPI) Flash Support

Diamond Programmer, combined with a Lattice cable fly-wire, supports the programming of Serial Peripheral Interface (SPI) flash devices. All Lattice FPGA devices supported by the Diamond software support SPI Flash.

Each Lattice FPGA device contains Lattice's Reveal IP. Diamond Programmer uses the JTAG interface to program the SPI flash device. Because of their bitstream compression capability, these FPGAs allow the use of smaller-capacity SPI memory devices.

Third Party SPI Flash Devices The following tables list supported third-party SPI flash devices. Click the third-party manufacturer name to link to the table of supported third-party SPI flash devices.

Note

The Diamond software Programmer support for additional third-party SPI flash devices may be added from time to time. To see if support for third-party SPI flash devices has been added, in addition to those listed in the tables below, check the SPI Flash options in the “Device Properties Dialog Box” on page 927.

Cypress

GigaDevice

ISSI

Macronix

Micron

Spansion

Winbond

Return to Top

Table 8: Cypress

S25FL116K
S25FL132K
S25FL164K
S25FL064L
S25FL128L
S25FL256L
S25FL128S
S25FL256S
S25FL512S
S25FS064S
S25FS128S
S25FS256S
S25FS512S

Return to Top

Table 9: GigaDevice

GD25Q20C
GD25Q40C
GD25Q80C
GD25Q16C
GD25Q32C
GD25Q64C
GD25Q127C
GD25Q256D
GD25LQ20C
GD25LQ40C
GD25LQ80C
GD25LQ16C
GD25LQ32D
GD25LQ64C
GD25LQ128D
GD25LQ256D

Return to Top

Table 10: ISSI

IS25LP064
IS25LP128
IS25LP032A
IS25LP064A
IS25LP080D
IS25LP016D
IS25LP032D
IS25LP256D
IS25LP128F
IS25LP512M
IS25LQ020B
IS25LQ040B
IS25LQ080B
IS25LQ016B
IS25LQ032B
IS25WP128
IS25WP064A
IS25WP040D
IS25WP080D
IS25WP016D
IS25WP032D
IS25WP128F
IS25WP256D
IS25WP512M
IS25WQ020
IS25WQ040

Return to Top

Table 11: Macronix

MX25L2006E	MX25R1035F
MX25L4006E	MX25R2035F
MX25L8006E	MX25V2035F
MX25L1606E	MX25R4035F
MX25L3206E	MX25U4035F
MX25L6406E	MX25V4035F
MX25U2033E	MX25R8035F
MX25U4033E	MX25V8035F
MX25U8033E	MX25R1635F
MX25L1635E	MX25U1635F
MX25U3235E	MX25V1635F
MX25L6435E	MX25R3235F
MX25U6435E	MX25U3235F
MX25L12835E	MX25R6435F
MX25L25635E	MX25U6435F
MX25L6436E	MX25L12835F
MX25L12836E	MX25U12835F
MX25L6445E	MX25L25635F
MX25L12845E	MX25L12845G
MX25U1633F	MX25L25645G
MX25L3233F	MX25U25645G
MX25L6433F	MX25L51245G
MX25L12833F	MX25U51245G
MX25V1035F	

Return to Top

Table 12: Micron

M25P10
M25P20
M25P40
M25P80
M25P16
M25P32
M25P64
M25P128
M25PE10
M25PE20
M25PE40
M25PE80
M25PE16
M45PE10
M45PE20
M45PE40
M45PE80
M45PE16
M25PX80
M25PX16
M25PX32
M25PX64
N25Q032
N25Q032A
N25Q064
N25Q128
N25Q128A
N25Q256
N25Q512
MT25QL128
MT25QL256
MT25QL512
MT25QU128
MT25QU256
MT25QU512

Return to Top

Table 13: Spansion

S25FL064A
S25FL040A
S25FL004A
S25FL008A
S25FL016A
S25FL032A
S25FL064A
S25FL001D
S25FL002D
S25FL004D
S25FL032P
S25FL064P
S25FL128P
S25FL132K
S25FL164K
S25FL204K
S25FL208K
S25FL216K

Return to Top

Table 14: Winbond

W25Q20
W25Q80
W25Q16
W25Q32
W25Q64
W25X10CL
W25X20CL
W25X40CL
W25Q20CL
W25Q40CL
W25Q80DV
W25Q32FV
W25Q64FV
W25Q128FV
W25Q256FV
W25Q16FW
W25Q32FW
W25Q64FW
W25Q128FW
W25Q16JV
W25Q32JV
W25Q64JV
W25Q128JV
W25Q256JV
W25M512JV
W25Q32JW
W25Q64JW
W25Q128JW
W25Q256JW

Third Party SPI Flash Devices (Legacy) The following tables list supported legacy third-party SPI flash devices. Click the third-party

manufacturer name to link to the table of supported legacy third-party SPI flash devices.

Note

The Diamond software Programmer support for additional legacy third-party SPI flash devices may be added from time to time. To see if support for third-party SPI flash devices has been added, in addition to those listed in the tables below, check the SPI Flash options in the "Device Properties Dialog Box" on page 927.

Adesto

AMIC

Atmel

Eon Silicon

Intel

Macronix

NewFlash

Numonyx

SSTI

STMicro

WindBond

.

Table 15: Adesto

AT25SF041

Return to Top of Legacy Device List

Table 16: AMIC

A25L10P
A25L20P
A25L40P
A25L80P
A25L16P

Return to Top of Legacy Device List

Table 17: Atmel

MX25L1005
MX25L2005
MX25L4005(A)
MX25L8005
MX25L1605
MX25L3205
MX25L6405
MX25L12805
MX25U8035

Return to Top of Legacy Device List

Table 18: Eon Silicon

EN25Q32B

Return to Top of Legacy Device List

Table 19: Intel

25F016S33
25F160S33
25F320S33
25F640S33

Return to Top of Legacy Device List

Table 20: Macronix

MX25L1005
MX25L2005
MX25L4005(A)
MX25L8005
MX25L1605
MX25L3205
MX25L6405
MX25L12805
MX25U8035

Return to Top of Legacy Device List

Table 21: NewFlash

NX25P20
NX25P40
NX25P80
NX25P16
NX25P32

Return to Top of Legacy Device List

Table 22: Numonyx

M25P20
M25P40
M25P80
M25P16
M25P32
M25P64
M25P128
M25PE20
M25PE40
M25PE80
M25PE16
M25PX80
M25PX16
M25PX32
M25PX64
M45PE20
M45PE40
M45PE80
M45PE16
N25Q032
N25Q064
N25Q128
N25Q128A

Return to Top of Legacy Device List

Table 23: SSTI

SST25VF020
SST25VF040
SST25VF080
SST25LF020A
SST25LF040A
SST25LF080A
SST25VF040B
SST25VF080B
SST25VF016B
SST25VF032B
SST25VF064C
SST26VF016B
SST26VF032B
SST26VF064B

Return to Top of Legacy Device List

Table 24: STMicro

M25P20
M25P40
M25P80
M25P16
M25P32
M25P64
M25PE20
M25PE40
M25PE80
M25PE16
M45PE20
M45PE40
M45PE80
M45PE16

Return to Top of Legacy Device List

Table 25: WindBond

W25P20
W25P40
W25P80
W25P16
W25P32
W25X10
W25X20
W25X40
W25X80
W25X16
W25X32
W25X64
W25Q128BV
W25Q64DW

Return to Top of Legacy Device List

See Also ▶"Programming the FPGA" on page 802

▶ "Using Diamond Programmer" on page 824

Programming Lattice Mature Devices with Stand-Alone Diamond Programmer

The stand-alone Diamond Programmer supports mature Lattice devices. A "Mature Device" license feature (LSC_PROGRAMMER_MATURE) is required in your license file to enable these devices. Without the license, Diamond Programmer will list the mature devices in the Device Selection and will be able to Scan the devices, but will not be able to program mature devices. The only operation available is Bypass. With the Mature Device license, full support is available. The Mature Device license feature is default in the free license. The free license can be requested from http://www.latticesemi.com/en/Support/Licensing.

Diamond Integrated Programmer (Programmer integrated into the Lattice Diamond software) will list the mature devices in the Device Selection and will be able to Scan the devices, but will not be able to program mature devices, even with the Mature Device license. The only operation available is Bypass.

Full support is only available when running Diamond Programmer as a stand-alone tool.

All devices listed below require a Mature Device license. Contact Lattice Technical Support to obtain a Mature Device license.

Table 26: Devices Supported in Stand-Alone Diamond Programmer That Require "Mature Device" License

Device Type	Device Family	Device Name
FPGA	ispXPGA	All
	ORCA	OR2T15A
		OR2T26A
		OR3T30
		OR3T55
		ORT4622
		ORSO42G5
		ORSO82G5
		ORT42G5
		ORT82G5
		ORT8850H
		ORT8850L
CPLD	ispXPLD	All
	ispMACH5000VG	All
	ispLSI 1000EA	All
	ispLSI 2000E / VE	All
	ispLSI 5000VA	All
	ispLSI 5000VE	All
	MACH4	M4-128N/64
	MACH4A	All
	MACH5	All
	MACH5/1	All
	MACH5LV	All
	MACH5-OVP / 5LV-OVP	All
Clock Manager	ispPAC-CLK5500	All
	ispPAC-CLK5600	All
Power Manager	ispPAC-POWR1208 / P1	All

Table 26: Devices Supported in Stand-Alone Diamond Programmer That Require "Mature Device" License

Device Type	Device Family	Device Name
Digital Interconnect	ispGDX2	LX64V / EV
		LX128V / EV
		LX256V / EV
	spGDXVA	All

See Also ▶"Programming the FPGA" on page 802

▶ "Using Diamond Programmer" on page 824

Programming Lattice Mature Devices with Stand-Alone Diamond Programmer

The stand-alone Diamond Programmer supports mature Lattice devices. A "Mature Device" license feature is required in your license file to enable these devices. Without the license, Diamond Programmer will list the mature devices in the Device Selection and will be able to scan the devices, but will not be able to program mature devices. The only operation available is Bypass. With the Mature Device license, full support is available. The Mature Device license feature is default in the free license. The free license can be requested at http://www.latticesemi.com/en/Support/Licensing.

Diamond Integrated Programmer (Programmer integrated into the Lattice Diamond software) will list the mature devices in the Device Selection and will be able to Scan the devices, but will not be able to program mature devices, even with the Mature Device license. The only operation available is Bypass.

Full support is only available when running Diamond Programmer as a stand-alone tool.

All devices listed below require a Mature Device license. Contact Lattice Technical Support to obtain a Mature Device license.

Table 27: Devices Supported in Stand-Alone Diamond Programmer That Require "Mature Device" License

Device Type	Device Family	Device Name
FPGA	ispXPGA	All
	ORCA	OR2T15A
		OR2T26A
		OR3T30
		OR3T55
		ORT4622
		ORSO42G5
		ORSO82G5
		ORT42G5
		ORT82G5
		ORT8850H
		ORT8850L
CPLD	ispXPLD	All
	ispMACH5000VG	All
	ispLSI 1000EA	All
	ispLSI 2000E / VE	All
	ispLSI 5000VA	All
	ispLSI 5000VE	All
	MACH4	M4-128N/64
	MACH4A	All
	MACH5	All
	MACH5/1	All
	MACH5LV	All
	MACH5-OVP / 5LV-OVP	All
Clock Manager	ispPAC-CLK5500	All
	ispPAC-CLK5600	All
Power Manager	ispPAC-POWR1208 / P1	All

Table 27: Devices Supported in Stand-Alone Diamond Programmer That Require "Mature Device" License

Device Type	Device Family	Device Name
Digital Interconnect	ispGDX2	LX64V / EV
		LX128V / EV
		LX256V / EV
	spGDXVA	All

See Also

- "Programming the FPGA" on page 802
- "Using Diamond Programmer" on page 824

Embedded Software Device Support

This topic describes device support for the following Lattice embedded software:

- "The JTAG Full VME Embedded Device Support" on page 820
- "JTAG Slim VME Embedded Device Support" on page 821
- "sysCONFIG Embedded Device Support" on page 822
- "Slave SPI Embedded Device Support" on page 823
- "I2C Embedded Programming Device Family Support" on page 823

The JTAG Full VME Embedded Device Support The JTAG Full VME Embedded software supports the following Lattice JTAG non-volatile devices. Lattice JTAG devices are those devices that can be programmed using the IEEE 1149.1 boundary scan TAP controller interface.

Table 28: Supported JTAG Full VME Embedded Devices

Device Type	Device
CPLD	ispMACH 4A
Electronically Erasable (EE) devices	ispXPGA
	ispXPLD
	ispGDX2
	ispGAL 22V10A
	ispMACH 5000B
	ispMACH 5000VG
	ispMACH 4000V/B/C/ZC

Table 28: Supported JTAG Full VME Embedded Devices

Device Type	Device
Flash Devices	LFMNX
	MachXO
	MachXO2
	MachXO3D
	MachXO3LF (not including rest of MachXO3L family of devices)
	LatticeXP/2
NVCM devices	MachXO3L (not including MachXO3LF)
FPGAs (Fast Program and SPI Flash Programming)	ECP5
	LatticeECP/EC
	LatticeECP2/M
	LatticeECP2S/MS
	LatticeSC
	LatticeECP3
Others	Power Manager
	Power Manager II
	Platform Manager
	Platform Manager 2
	ispCLOCK

Return to Top

JTAG Slim VME Embedded Device Support The JTAG Slim VME Embedded is designed for 8051 micro controller applications where resources are limited. It takes advantage of the much-simplified programming algorithm adopted by the following Lattice's IEEE 1532-compliant PLD families.

Table 29: JTAG Slim VME Embedded Device Support

Device Type	Device
CPLD	ispMACH 4A
Electronically Erasable (EE) devices	ispXPGA
	ispXPLD
	ispGDX2
	ispGAL 22V10A
	ispMACH 5000B
	ispMACH 5000VG
	ispMACH 4000V/B/C/ZC

Table 29: JTAG Slim VME Embedded Device Support

Device Type	Device
Flash Devices	LFMNX
	MachXO
	MachXO2
	MachXO3D
	MachXO3LF (not including rest of MachXO3L family of devices)
	LatticeXP/2
NVCM devices	MachXO3L (not including MachXO3LF)
FPGAs (Fast Program and SPI Flash Programming)	ECP5
	LatticeECP/EC
	LatticeECP2/M
	LatticeECP2S/MS
	LatticeSC
	LatticeECP3
Others	Power Manager
	Power Manager II
	Platform Manager
	Platform Manager 2
	ispCLOCK

Return to Top

sysCONFIG Embedded Device Support The sysCONFIG Embedded software supports the following Lattice volatile devices, using sysCONFIG as the target port.

Table 30: Supported sysCONFIG Embedded Devices

Device Type	Device
XPLD	ispXPLD 5000MX
Flash devices	LatticeXP
FPGAs	ECP5
	LatticeECP/EC
	LatticeECP2/M
	LatticeECP2S/MS
	LatticeECP3
	LIFMD
	LIFMDF

Return to Top

Slave SPI Embedded Device Support The Slave SPI Embedded software supports the following Lattice devices.

Table 31: Supported Slave SPI Embedded Devices

Device Type	Device
Flash devices	LFMNX
	MachXO2
	MachXO3D
	MachXO3L
	MachXO3LF (not including rest of MachXO3L family of devices)
	Platform Manager 2
	LatticeXP/2
	LIFMDF
NVCM devices	MachXO3L (not including MachXO3LF)
FPGAs	ECP5
	LatticeECP3
	LIFMD

Return to Top

I^2C Embedded Programming Device Family Support The I^2C Embedded software supports Lattice devices listed below.

Table 32: Supported I^2C Embedded Programming Devices

Device Type	Device
I^2C embedded programming devices	LFMNX
	MachXO2
	MachXO3D
	MachXO3L
	Platform Manger 2
	LIFMD
	LIFMDF

Return to Top

Using Diamond Programmer

This section provides procedures for using Diamond Programmer. Topics include:

- "Installing/Uninstalling Parallel Port Driver and USB Driver on a PC" on page 825
- "Troubleshooting the USB Driver on a PC" on page 826
- "Installing and Configuring USB Cable and Parallel Cable on Linux" on page 827
- "Plugging the Cable into the PC" on page 827
- "Detecting a Cable" on page 827
- "Selecting from Multiple Cables" on page 828
- "Creating a New Diamond Programmer Project" on page 829
- "Opening an Existing Diamond Programmer Project" on page 830
- "Checking the XCF Project" on page 831
- "Scanning the Device Chain" on page 832
- "Selecting from a Matching Device ID List" on page 832
- "Adding a Lattice Device to a Chain" on page 833
- "Adding a Generic JTAG Device to a Chain" on page 833
- "Removing a Device from a Chain" on page 834
- "Editing Device Properties" on page 834
- "Moving a Device Up or Down" on page 835
- "Setting Chain Operations" on page 835
- "Setting Programming Characteristics" on page 835
- "Downloading Design Files" on page 836
- "Editing the I/O State" on page 836
- "Adding a Custom SPI Flash Device" on page 837
- "Editing a Custom SPI Flash Device" on page 838
- "Removing a Custom SPI Flash Device" on page 839
- "Programming Using Custom SPI Flash Device" on page 839
- "Setting up Dual Boot Golden Sector Protection Support" on page 840
- "Programming LFMNX, MachXO2, MachXO3D, and MachXO3L Devices Using I2C" on page 841
- "Editing the Security Feature Row (MachXO2, MachXO3L/LF, MachXO3D, and LFMNX Only)" on page 843
- "Configuring SPI Flash Options" on page 852
- "Configuring Advanced Flash Programming Options" on page 852
- "Configuring the BSCAN2" on page 853

- ""Programming the FPGA" on page 802Changing the Port Assignment" on page 854
- "Testing the Cable Signal" on page 855
- "Viewing the Log File" on page 855

Installing/Uninstalling Parallel Port Driver and USB Driver on a PC

Parallel port or USB drivers are required to program Lattice devices using the Lattice download cables and Diamond Programmer on a PC. To install the drivers, you should have administrative privileges.

Three drivers can be installed or uninstalled with the LSC Drivers Install/ Uninstall dialog box.

- **Parallel port driver** – Supports device programming through the parallel port of your PC. The driver can be installed on 64-bit Windows operating systems.
- **USB port driver** – Supports device programming through the USB port of your PC. The driver can be installed on Windows 8 and 10 (64-bit).
- **FTDI USB driver** – Supports Lattice evaluation boards with FTDI (Future Technology Devices International) USB host chip. The driver does not support Windows 2000 and earlier operating systems.

To install the driver during software installation:

1. In the LSC Drivers Install/Uninstall dialog box, select one driver or **All Drivers** to be installed.
2. If another driver is already installed, you can click **Uninstall** if you want to remove it.
3. Click **Install**.
4. Click **Close**, and follow the installation instructions on the screen.

To install the driver after Diamond has been installed:

1. In Windows choose **Programs > Lattice Diamond > Accessories > Install & Uninstall Cable Drivers**.
2. In the LSC Drivers Install/Uninstall dialog box, select one driver or **All Drivers** to be installed.
3. If another driver is already installed, you can click **Uninstall** if you want to remove it.
4. Click **Install**.
5. Click **Close**, and follow the installation instructions on the screen.

See Also ▶"Using Diamond Programmer" on page 824

▶ "Programming the FPGA" on page 802

Troubleshooting the USB Driver on a PC

If you meet problems with the USB driver, try the troubleshooting procedure as follows:

1. Uninstall the USB driver. Refer to "Installing/Uninstalling Parallel Port Driver and USB Driver on a PC" on page 825.
2. Disconnect any USB devices or cables from your PC.
3. Reboot your PC.
4. Install the LSC Windows USB driver following the steps listed above.
5. Reboot the PC.
6. After the PC boots, plug a USB cable into your PC.

   Your PC should automatically detect the cable and install the correct driver.
7. Open the Device Manager.

   You should see Lattice USB Programmer under Universal Serial Bus controllers.
8. Right-click Universal Serial Bus controllers and choose **Properties**.

   In the Drivers tab, you should see "Driver Provider: %Tril%". If not, the driver is not installed properly. Click the Uninstall button and go back to step 2.
9. Launch Diamond Programmer. In the Cable and I/O Settings tab, click **Detect Cable**.

   The Diamond Programmer should detect the USB cable as EzUSB-0.
10. Connect the cable to the target board, choose **Design > JTAG Scan** or click  in the toolbar.

    The Diamond Programmer should scan the JTAG devices on the board.

See Also ▶"Using Diamond Programmer" on page 824

▶ "Programming the FPGA" on page 802

Installing and Configuring USB Cable and Parallel Cable on Linux

For information on installing and configuring USB and Parallel cables on Linux, refer to the *Lattice Diamond Installation Notice for Linux*.

See Also ▶"Using Diamond Programmer" on page 824

▶ "Programming the FPGA" on page 802

Plugging the Cable into the PC

You can plug the USB cable into the PC when the PC is turned off or when it is on. Make sure that the Diamond Programmer is closed before plugging the cable into your PC or unplugging it from your PC.

To plug the cable into the PC:

1. Make sure you have installed the USB port driver. See "Installing/ Uninstalling Parallel Port Driver and USB Driver on a PC" on page 825 for details.
2. Make sure that Diamond Programmer is closed, and then plug the cable into your PC.
3. If your PC is turned on, wait about one minute for the Windows operating system to recognize the USB cable. The amount of time will vary, depending on your PC's speed.

Note

Be sure to turn off the target board's power before connecting or disconnecting the USB cable to the target board.

See Also

▶ "Using Diamond Programmer" on page 824

▶ "Programming the FPGA" on page 802

Detecting a Cable

You can use the Diamond Programmer Detect Cable feature to determine which cable and port you are using.

To detect a cable:

- In the "Cable Settings Tab" on page 944, click **Detect Cable**. The Diamond Programmer detects all available cables connected to your PC, and lists the cable type, port settings, and descriptions in the Output console.

To detect or upgrade an FTDI cable:

1. To plug in FTDI cable, see "Plugging the Cable into the PC" on page 827.
2. In the "Cable Settings Tab" on page 944, click **Detect Cable**. The Diamond Programmer detects the HW-USBN-2B (FTDI) cable connected to your PC, FTUSB-0 as the port settings, and descriptions in the Output console.
3. Click on "Firmware Tab" on page 946. A new window displays fields for MachXO2 firmware and POWR607 firmware.
4. Select a JEDEC (*.jed) file to update device/devices.

   Note

   If firmware versions of two JEDEC files don't match, the software will display a warning message asking if you want to continue.

5. Click on **Update Firmware** to reprogram devices on cable to specified file.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Selecting from Multiple Cables

Diamond Programmer can recognize as many as five USB cables plugged into the same PC.

The first USB cable plugged into the PC is assigned port Ez-USB-0. The second is assigned port Ez-USB-1, and so on. The first FTDI USB2 cable is assigned port FTUSB-0, and the second FTUSB-1, and so on.

If the PC is turned off, and then later turned back on, the assigned ports might change, since the PC detects the cables according to the USB port address instead of the order in which they were plugged into the PC. Since the USB cable port assignments might change during power-up, you might have to reselect the USB cable in the "Cable Settings Tab" on page 944.

To select a USB cable from multiple cables:

- In the "Cable Settings Tab" on page 944, select the USB cable you want to use from the Port drop-down list.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Creating a New Diamond Programmer Project

By creating a new Diamond Programmer project, you create a chain file (.xcf) from a scan with default settings, create an .xcf file from a scan with custom settings, or a new blank project.

How you start Programmer also depends on whether you are using it integrated with Diamond or using the stand-alone version, and on your operating system.

Note

You can change the way Diamond Programmer writes the paths to data files in the .xcf file by editing the Diamond Programmer .ini file.

By default, the path in the .xcf file is displayed in “cross platform compatible” format, meaning forward-slashes (/) when using Diamond Programmer on Windows and Linux. By editing the programmer .ini file, you can change the way the path is saved to Native Delimiter format, meaning back-slashes (\) when using Diamond Programmer on Windows. Diamond Programmer on Linux will continue to use forward-slashes (/), thus, this option is not required for Linux.

To enable Native Delimiter format, add the following line to the end of the programmer.ini file:

PathDelimiter=1

If you wish to change back to the default Cross Platform Compatible format, you can change the value in the programmer.ini file to:

PathDelimiter=0

or delete the line entirely.

The programmer.ini file is located in the following directory:

<*Drive*>:\Users\<*User Name*>\AppData\Roaming\LatticeSemi

To create a new Diamond Programmer project:

1. Make sure that the board is turned on and that the Lattice parallel or USB cable is properly connected to the computer.
2. Issue the start command.
   - In the Diamond main window, choose **Tools > Programmer**; or in Windows choose **Programs > Lattice Diamond > Accessories > Programmer**; or click  in the Diamond toolbar.

- In Linux, from the <*install_path*>/bin/lin directory, enter the following on a command line:

  ```
  ./programmer
  ```

3. In the "Getting Started Dialog Box" on page 856, choose one of the following:
   - **Create a New Diamond Programmer Project from a JTAG Scan.** This option creates a new project from a scan.
     - If you have a board connected to your computer, you can click the **Detect Cable** button to detect the cable and port. Otherwise, you can manually select cable and port from the **Cable** and **Port** pull-down lists.
   - **Create a New Blank Project**—This option creates a new blank programmer project.

   Note

   If you wish to import the .xcf file into your Diamond project, choose **Import new Programmer project to current implementation**.

4. Click **OK**.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Opening an Existing Diamond Programmer Project

To open an existing Diamond Programmer project, you must have an existing chain file (.xcf). The daisy chain configuration must be the same as that for which the chain configuration file was created. Each unique chain configuration requires a unique chain configuration file.

How you start Programmer also depends on whether you are using it integrated with Diamond or using the stand-alone version, and on your operating system.

To start Diamond Programmer with an existing project:

1. Issue the start command. To start:
   - In the Diamond main window, choose **Tools > Programmer**.
   - The stand-alone Diamond Programmer in Windows, go to the Windows Start menu and choose **Programs > Lattice Diamond > Accessories > Programmer**

- The stand-alone Diamond Programmer in Linux, enter the following on a command line:

  ```
  /.programmer
  ```

2. In the "Getting Started Dialog Box" on page 856, choose **Open an Existing Diamond Programmer Project.**
   - **Browse** to the desired .xcf file.

   Note

   If you wish to import the .xcf file into your Diamond project, choose **Import new Programmer project to current implementation**.

3. Click **OK**.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Checking the XCF Project

The Check XCF button allows you to perform a design rule check on your XCF project setup before performing any actions. This feature is run automatically when the .xcf file is saved.

To check the XCF project:

- In Diamond Programmer, choose **Design > Check XCF Project** or click  in the toolbar. Diamond Programmer checks the XCF project and indicates in the output pane whether or not the project is valid.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Scanning the Device Chain

There are several ways to scan a device chain. When you create a new project, you can choose to scan you board, perform a custom scan, or create a blank project without performing a scan.

Note

Not all Lattice device families support Scan capability. Refer to the *iCEcube2 Release Notes* for more information on the device family that does not support Scan.

To scan the device chain on the board that is connected to your PC:

1. Make sure that the board is turned on and that the cable is properly connected to Diamond Programmer.
2. In Diamond Programmer, choose **Design > JTAG Scan** or click  in the toolbar. Diamond Programmer opens a new chain configuration file, scans the printed circuit board connected to your computer, and lists the devices in the new chain file.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Selecting from a Matching Device ID List

When a scanned device shares the same device ID with other devices, the software indicates this in the main window with the message “Cannot identify detected device on row <*row number*>. Please manually select correct device.”

To select a different device from the matching ID list:

1. Click the device in the Device column
2. Select the device you want to use from the dropdown menu.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Adding a Lattice Device to a Chain

You can add any Lattice device to an existing chain.

To add a Lattice device to a chain:

1. In Diamond Programmer, choose **Edit > Add Device,** or click the button on the toolbar, or right-click and choose **Add Device**.
2. Select a Lattice device family from the **Device Family** column dropdown menu.
3. Select a device the **Device** column dropdown menu.

   The software inserts the new device into the active chain.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Adding a Generic JTAG Device to a Chain

You can add any one of the following types of devices to a chain:

- JTAG-NOP Device
- JTAG-SVF device that has a serial vector format file (.svf)
- JTAG-ISC
- JTAG-STAPL

To add a Generic JTAG device:

1. In Diamond Programmer, choose **Edit > Add Device,** or click the button on the toolbar or right-click and choose **Add Device**.
2. Select Generic JTAG Device from the **Device Family** column dropdown menu.
3. Select a either JTAG-NOP, JTAG-SVF, JTAG-ISC, or JTAG-STAPL from the **Device** column dropdown menu.
4. Highlight the row, and choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties**. In the "Device Properties Dialog Box—Device, Access Mode, and Operation Options" on page 857, click the Browse button and navigate to the .jed, .bit, .isc, or .rbt file for the device. Select the file and click **Open**.

   The software inserts the new device into the active chain.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Removing a Device from a Chain

To remove a device from the active chain:

1. In Diamond Programmer, select the device that you want to delete.
2. Choose **Edit > Remove Device**, or click the  button on the toolbar, or right-click and chose **Remove Device**.

Note

There is no Undo for this command.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Editing Device Properties

To edit device properties in a chain:

1. In Diamond Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the  button on the toolbar, or right-click and choose **Device Properties**.
3. In the “Device Properties Dialog Box” on page 927, edit the device properties.
4. Click **OK** to close the “Device Properties Dialog Box” on page 927.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Moving a Device Up or Down

To move a device up or down:

- In Diamond Programmer, with the left mouse button, select and hold the number in the far left in the row of the device that you want to move. Then drag and drop to the row where you want to move the device.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Setting Chain Operations

To set a chain operation in a device:

1. Select a device in the chain.
2. Double-click the **Operation** column for the device.
3. In the "Device Properties Dialog Box" on page 927, select the specific operation for the device in the Operation drop down menu.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Setting Programming Characteristics

Use the "Settings Dialog Box" on page 943 to specify how Diamond Programmer will process the active chain configuration: process the whole chain or only a single device; override selected operations of the selected device or all the devices in the chain; process the chain in sequential or turbo mode. You can also specify other options.

To specify Programming characteristics:

1. In Diamond Programmer, choose **Edit > Settings**.

   The "Settings Dialog Box" on page 943 opens.
2. Select the options you want, and then click **OK**.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Downloading Design Files

After you have specified all of the processing options, use the Program command to download the design files to the daisy-chained devices.

- In Diamond Programmer, choose **Design > Program,** or click  on the toolbar.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Editing the I/O State

Editing I/O state allows you to change values of configurable pins to specified values by using the HighZ, All 1s, All 0s, Leave Alone, Custom, and Dynamic I/O options.

Controlling the I/O state allows you to edit the state of any of the I/O pins before programming.

States include:

- **HighZ**—Sets all pins to tristate. This is the default setting.
- **All 1’s—**Preloads the boundary scan register with all 1’s.
- **All 0’s—**Preloads the boundary scan register with all 0’s.
- **Leave alone**—Uses the current values for all pins throughout the programming cycle.
- **Custom**—Uses the specified pin settings from the “Edit I/O State Dialog Box” on page 947.
- **Dynamic I/O**—Custom leave-alone option that enables you to specify the setting for selected pins in the boundary scan register before programming and leave all other pins at their current state. When Dynamic I/O is selected, the “X” in the “Edit I/O State Dialog Box” on page 947 is Leave Alone.

To edit I/O state:

1. Right-click on the row corresponding to the device you would like to edit.
2. Choose **Edit I/O State**.

3. In the "Edit I/O State Dialog Box" on page 947, choose the desired I/O state option from the I/O State dropdown.
4. In the BSDL File box, browse to the appropriate BSDL file for the device that you are editing.
5. For each pin that you want to change, click the pin's square repeatedly until it displays the value you want.
6. Right-click the pin's square and select the value you want from the pop-up menu.
7. Click **OK**.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Adding a Custom SPI Flash Device

This feature allows you users to add your own SPI Flash device to the Diamond Programmer database.

To create a Custom SPI Flash device:

1. Right-click on the row corresponding to the device to which you would like to add a Custom SPI Flash device, and choose **Edit > Custom Device**.
2. In the "Edit Custom Device Dialog Box" on page 941, select **SPI Serial Flash Custom**, and click **Add**.
3. In the "Custom SPI Flash Dialog Box" on page 942, enter:
   - **Device Description**—this can be any alpha-numeric string that describes the device (for example: M25P32-VMF6C).
   - **Device Name**—this can be any alpha-numeric string that describes the device name (for example: SPI-M25P32).
   - **Package—**this can be any alpha-numeric string that describes the package (for example: 16-pin SOIC).
   - **Device ID—**this can be any alpha-numeric string that describes the device ID (for example: 0x15).
4. In the Device Vendor dropdown menu, choose device vendor.
5. In the Device Density dropdown, choose device density.
6. In the Byte Per Sector dropdown, choose the desired value.
7. If the SPI Flash device supports sector protection, check the **Protection Options ON** box.
8. Click **OK**.

9. Once the custom SPI flash has been added, it can be selected in the "Device Properties Dialog Box" on page 927.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Editing a Custom SPI Flash Device

If you have already created a custom SPI Flash device, you can edit its properties in the "Custom SPI Flash Dialog Box" on page 942.

To edit the properties of a Custom SPI Flash device:

1. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Custom Device**.
2. In the "Edit Custom Device Dialog Box" on page 941, select **SPI Serial Flash Custom**, select the SPI Flash device you want to edit, and click **Edit**.
3. In the "Custom SPI Flash Dialog Box" on page 942, enter:
   - **Device Description**—this can be any alpha-numeric string that describes the device (for example: M25P32-VMF6C).
   - **Device Name**—this can be any alpha-numeric string that describes the device name (for example: SPI-M25P32).
   - **Package**—this can be any alpha-numeric string that describes the package (for example: 16-pin SOIC).
   - **Device ID**—this can be any alpha-numeric string that describes the device ID (for example: 0x15).
4. In the Device Vendor dropdown menu, choose device vendor.
5. In the Device Density dropdown, choose device density.
6. In the Byte Per Sector dropdown, choose the desired value.
7. If the SPI Flash device supports sector protection, check the **Protection Options ON** box.
8. Click **OK**.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Removing a Custom SPI Flash Device

f you have already created a custom SPI Flash device, and you wish to remove it from your the “Custom SPI Flash Dialog Box” on page 942.

To edit the properties of a Custom SPI Flash device:

1. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Custom Device**.
2. In the “Edit Custom Device Dialog Box” on page 941, select **SPI Serial Flash Custom**, select the SPI Flash device you want to remove, and click **Remove** and **OK**.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Programming Using Custom SPI Flash Device

After a custom SPI Flash device has been created, you can program it using Diamond Programmer.

To program using a Custom SPI Flash device:

1. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Device Properties** to display the “Device Properties Dialog Box” on page 927.
2. In the Access Mode dropdown list, choose **SPI Flash Programming**.
3. In the Family dropdown list, choose **SPI Serial Flash Custom**.
4. In the Device dropdown list, choose the name of the custom device you created.
5. Click **OK.**

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Setting up Dual Boot Golden Sector Protection Support

Dual boot sector support allows for write protecting sectors containing golden pattern, and prevents golden pattern from being accidentally erased or programmed.

To enable Dual Boot Golden Sector Protection support:

1. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Device Properties**.
2. In the Access Mode dropdown list, choose **SPI Flash Programming**.
3. In the Operation dropdown list, chose desired operation.
4. In the Family dropdown list, choose **SPI Serial Flash**.
5. In the Vendor list, choose the name of the vendor.
6. In the Device dropdown list, choose the device.
7. In the Package dropdown list, choose package.
8. If the SPI flash device supports protection, the **Secure SPI Flash Golden Pattern Sectors** box should be enabled. If you wish to secure the golden sector, check this box.
9. Click **OK.**

The following Lattice Technical Notes give device-specific information on using the dual boot feature:

- FPGA-TN-02069, MachXO3D Programming and Configuration Usage Guide
- TN1279, MachXO3L Programming and Configuration Usage Guide
- TN1204, MachXO2 Programming and Configuration Usage Guide
- TN1216, LatticeECP2/M and LatticeECP3 Dual Boot Feature
- TN1220, LatticeXP2 Dual Boot Feature

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Unlocking Secure SPI Flash Support

If a custom SPI flash device or a SPI flash device that supports secure golden patterns sectors was secured as described in “Setting up Dual Boot Golden Sector Protection Support” on page 840, and you wish to reprogram the SPI flash, you must first unlock the device before reprogramming.

To unlock a secure SPI flash:

1. Create a new Diamond Programmer project as described in “Creating a New Diamond Programmer Project” on page 829 or open an existing Diamond Programmer project as described in “Opening an Existing Diamond Programmer Project” on page 830.
2. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Device Properties**.
3. In the Operation dropdown list, chose **SPI Flash Unlock Device**.
4. Ensure that the **Secure SPI Flash Golden Pattern Sectors** box is unchecked.
5. Click **OK.**
6. In Diamond Programmer, choose **Design > Program,** or click  on the toolbar.

The SPI flash is now unlocked and can be reprogrammed.

The following Lattice Technical Notes give device-specific information on using the dual boot feature:

- FPGA-TN-02069, MachXO3D Programming and Configuration Usage Guide
- TN1279, MachXO3L Programming and Configuration Usage Guide
- TN1204, MachXO2 Programming and Configuration Usage Guide
- TN1216, LatticeECP2/M and LatticeECP3 Dual Boot Feature
- TN1220, LatticeXP2 Dual Boot Feature

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Programming LFMNX, MachXO2, MachXO3D, and MachXO3L Devices Using I2C

Diamond Programmer supports I2C programming of LFMNX, MachXO2, MachXO3D, and MachXO3L devices. Diamond Programmer requires an FTDI 2232H chip installed on the customer board to bridge the Lattice HW-USBN-2B (FTDI) USB programming cable to the LFMNX, MachXO2, MachXO3D, and MachXO3L I2C sysCONFIG port. For design reference, both the Lattice MachXO2 Pico Evaluation Board and the Lattice MachXO2 Control Evaluation Board contain working examples of the FTDI circuit.

To program a LFMNX or MachXO3D using I2C:

1. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Device Properties**.
2. In the Port Interface dropdown list, choose **I2C Interface**.
3. In the Operation dropdown list, choose desired operation.
4. Enter the I2C slave address of the LFMNX or MachXO3D device.
5. Click **OK**.
6. Download the design file.

To program a MachXO2 or MachXO3L using I2C:

1. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Device Properties**.
2. In the Access Mode dropdown list, choose **I2C Interface Programming**.
3. In the Operation dropdown list, choose desired operation.
4. Enter the I2C slave address of the MachXO2 or MachXO3L device.
5. Click **OK**.
6. Download the design file.

There is also an embedded solution which included in the *<install_path>**<version_number>*\embedded_source\i2cembedded directory. The embedded solution can be modified to target your processor or to use something other than the Future device.

For more information on the MachXO2 and MachXO3L I2C programming refer to TN1204, *MachXO2 Programming and Configuration Usage Guide* or TN1279, *MachXO3L Programming and Configuration Usage Guide*.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Programming Platform Manager 2, MachXO2, MachXO3D, or MachXO3L/LF Devices Using I2C with External ASC

Diamond Programmer supports I2C programming of Platform Manager 2, MachXO2, MachXO3D, or MachXO3L/LF devices with external ASC. Diamond Programmer requires an FTDI 2232H chip installed on the customer board to bridge the Lattice HW-USBN-2B (FTDI) USB programming cable to the Platform Manager 2, MachXO2, MachXO3D, or MachXO3L/LF and ASC I2C port.

To program a Platform Manager 2, MachXO2, MachXO3D, or MachXO3L/LF Device Using I2C with External ASC:

1. Right-click on the row corresponding to the device you would like to edit, and choose **Edit > Device Properties**.
2. In the Access Mode dropdown list, choose **I2C Interface Programming**.
3. In the Operation dropdown list, choose desired operation.
4. Enter the I2C slave address of the Platform Manager 2, MachXO2, MachXO3D, or MachXO3L/LF device.
5. In the External ASC Options box, click + to add an external ASC / LPTM21L device.
6. Select the data file, operation, and I2C slave address.
7. Repeat steps 5 and 6 for each external ASC / LPTM21L device.
8. Click **OK**.
9. Download the design file.

See Also

- “Using Diamond Programmer” on page 824
- “Programming the FPGA” on page 802

Editing the Security Feature Row (MachXO2, MachXO3L/LF, MachXO3D, and LFMNX Only)

You can read the contents of the feature row of the MachXO2, MachXO3L/LF, MachXO3D, or LFMNX, edit the settings, and program the new settings into the feature row of the device.

To Read the Feature Row of a MachXO2, MachXO3L/LF, MachXO3D, or LFMNX Device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the “Device Properties Dialog Box” on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.
4. In the Operation box, select **Security Read Feature Row**.
5. Click **OK** to close the Device Properties dialog box.
6. Choose **Design > Program,** or click on the toolbar.

The Feature Row dialog box displays. The fields in the dialog box are not editable.

7. Click **Close** to close the Feature Row dialog box.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Reading the Security Feature Row (MachXO2, MachXO3D, MachX03L/LF, LIFMDF, and LFMNX Only)

You can read the contents of the feature row of the LFMNX, MachXO3D or LIFMDF, edit the settings, and program the new settings into the feature row of the device.

To read the feature row of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Feature Row Programming**.
4. In the Operation box, select **Read Feature Row**.
5. Click **OK** to close the Device Properties dialog box.
6. Choose **Design > Program,** or click on the toolbar.

   The Feature Row dialog box displays. The fields in the dialog box are not editable.
7. Click **Close** to close the Feature Row dialog box.

To read the non-volatile control register 1 of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose Device Properties to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Feature Row Programming**.

4. In the Operation box, select **Display Control NV Register1**.
5. Click **OK** to close the Device Properties dialog box
6. Choose **Design > Program,** or click on the toolbar.

   The non-volatile control register 1 dialog box displays. The fields in the dialog box are not editable.
7. Click **Close** to close the dialog box.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Programming the Feature Row (MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, and LFMNX Only)

You can program the contents of a .fea file to the feature row of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device.

To erase the feature row of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Feature Row Programming**.
4. In the Operation box, select **Erase Feature Row**.
5. Click **OK** to close the Device Properties dialog box.
6. Choose **Design > Program,** or click on the toolbar.

To program the feature row of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Feature Row Programming**.

4. In the Operation box, select **Program Feature Row**.
5. In the Feature Row Programming Options box, browse to the Feature Row (.fea) file.
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

To edit the feature row of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Feature Row Programming**.
4. In the Operation box, select **Update Feature Row**.
5. Click **OK** to close the Device Properties dialog box.
6. Choose **Design > Program,** or click on the toolbar.

   The Feature Row dialog box displays.
7. Edit the fields in the dialog box as desired, then click **Program**. A dialog box asking if you wish to overwrite the selected register. If you click Yes, the LFMNX, MachXO3D, or LIFMDF device feature row will be programed as edited.

To edit the Non-Volatile Control Register 1 of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Feature Row Programming**.
4. In the Operation box, select **Program Control NV Register1**.
5. Click **OK** to close the Device Properties dialog box.
6. Choose **Design > Program,** or click on the toolbar.
7. In the non-volatile Control Register 1 dialog box displays.
8. Edit the fields in the dialog box as desired, then click Program. A dialog box asking if you wish to overwrite the selected register. If you click Yes, the MachXO3D device non-volatile Control Register 1 will be programed as edited.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Programming the Password Key (MachXO2, MachXO3D, MachX03L/LF, LIFMDF, and LFMNX Only)

You can program the password key from a .key file to the LFMNX, MachXO3D, and LIFMDF devices.

To read the Password key of a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.
4. In the Operation box, select **Security Read Password Key.**
5. Click OK to close the Device Properties dialog box
6. Choose **Design > Program,** or click on the toolbar.

To program the Password Key to a MachXO2, MachXO3D, MachXO3L/LF, LIFMDF, or LFMNX device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.
4. In the Operation box, select **Security Program Password Key.**
5. In the Password Key Programming Options box, browse to the key (.key) file.
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

Programming the Encryption Key (LFMNX and MachXO3D Only)

You can program the encryption key from a .bek file to the LFMNX and MachXO3D device.

To read the Encryption key of a LFMNX and MachXO3D Device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.
4. In the Operation box, select **Security Read Encryption Key**.
5. Click **OK** to close the Device Properties dialog box
6. Choose **Design > Program,** or click on the toolbar.

To program the Encryption Key to a LFMNX and MachXO3D Device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.
4. In the Operation box, select **Security Program Encryption Key.**
5. In the Encryption Key Programming Options box, browse to the key (.bek) file.
6. [Optional] Select **Read Lock** and/or **Write Lock** based on choice of security to lock the encryption key register.
7. Click **OK** to close the Device Properties dialog box.
8. Choose **Design > Program,** or click on the toolbar.

Programming the Public Key (LFMNX and MachXO3D Only)

You can program the public key from a .pub file to the LFMNX and MachXO3D device.

To read the Public key of a LFMNX or MachXO3D Device:

1. In Programmer, select the device that you want to edit.

2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.
4. In the Operation box, select **Security Read Public Key.**
5. Click **OK** to close the Device Properties dialog box.
6. Choose **Design > Program,** or click on the toolbar.

To program the Public Key to a LFMNX or MachXO3D Device:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Security Keys Programming**.
4. In the Operation box, select **Security Program Public Key**.
5. In the Public Key Programming Options box, browse to the key (.pub) file.
6. Click **OK** to close the Device Properties dialog box.
7. Choose **Design > Program,** or click on the toolbar.

Setting LFMNX and MachXO3D Dry Run Features

The LFMNX and MachXO3D device families can trigger a dry run bitstream downloading from either on-die flash or external flash through the master SPI port. "Dry run" means downloading the bitstream but without actually writing the configuration SRAM.

The following are done during the "dry run" process:

- Decrypting the Bitstream if it is encrypted
- Authentication with the Bitstream if it is required
- CRC check if it is enabled
- Execute command "ISC_PROGRAM_USERCODE" inside the bitstream and store the USERCODE to a special shadow register named "DRYRUN USERCODE". The command will not override "SRAM USERCODE" nor "Flash USERCODE".

After the "dry run" finished it will report the execution status, such as execution pass/fail and authentication pass/fail. The user can use the command "USERCODE_DRYRUN" to read the shadow register of "DRYRUN USERCODE".

To execute a dry run from external SPI Flash Primary Image in Programmer:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Static RAM Cell Mode**.
4. In the Port Interface drop-down menu, choose the interface.
5. In the Operation box, select **SRAM External Primary Dry Run**.
6. Click **OK**.
7. Choose **Design > Program,** or click on the toolbar.

   The USERCODE from the Primary image will displayed if the "Dry Run" executed successfully.

To execute a dry run from external SPI Flash Golden Image in Programmer:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Static RAM Cell Mode**.
4. In the Port Interface drop-down menu, choose the interface.
5. In the Operation box, select **SRAM External Golden Dry Run**.
6. Click **OK**.
7. Choose **Design > Program,** or click on the toolbar.

   The USERCODE from the Golden image will displayed if the "Dry Run" executed successfully.

To execute a dry run from on-die Flash in Programmer:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Flash Programming Mode**.
4. In the **Port Interface** drop-down menu, choose the interface.
5. In the Operation box, select **Flash Dry Run**.
6. Check the Flash-A and/or Flash-B box desired for "dry run."
7. Click **OK**.

8. Choose **Design > Program,** or click on the toolbar.

   The USERCODE from the Flash-A and/or Flash-B will displayed if the "Dry Run" executed successfully.

Setting LFMNX and MachXO3D Version Rollback Protection Features

The LFMNX and MachXO3D device families can support version rollback downloading from either on-die flash or external flash through the master SPI port. "Version Rollback" means downloading first version or the latest version bitstream depending on user settings.

The following are done during the "Version Rollback" process:

- Decrypting the Bitstream if it is encrypted
- Authentication with the Bitstream. It is always required for this feature.
- CRC check if it is enabled
- Program the Authenticated DONE
- Execute command "ISC_PROGRAM_USERCODE" inside the bitstream
- Compare the USERCODE time stamp of two FLASH Blocks
- Booting from the FLASH sector that had the first version or the latest version bitstream depending on user settings

After the "Version Rollback" finished it will report the execution status, such as execuction pass/fail and authentication pass/fail. The user can use the command "USERCODE" to read the USERCODE time stamp.

Note

The VRBP_EN bit in the feature row need to set to 1 to enable the version rollback protection feature.

To execute a "Version Rollback" testing from on-die Flash in Programmer:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Flash Programming Mode**.
4. In the Port Interface drop-down menu, choose the interface.
5. In the Operation box, select **Flash Version Rollback Protection**.
6. Provide the data files (jed) for both Flash A and Flash B.
7. Click **OK**.

8. Choose **Design > Program,** or click on the toolbar.

   The USERCODE of the first version or the latest version bitstream depending on user settings will be displayed.

Configuring SPI Flash Options

You can use the "Device Properties Dialog Box" on page 927 to configure SPI Flash options for LatticeXP2 devices.

To configure SPI Flash Options:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **SPI Flash Programming**.
4. In the Options drop-down menu, chose the desired SPI Flash option.
5. Select **Family**, **Vendor**, **Device**, and **Package**.
6. Click **Load Size from Programming File** to fill in the Data File Size box. You can change this size by typing a different file size. This feature is useful if you only want to use a portion of the file for the operation. If you change the file size, it must be no larger than the full file size or the device size.
7. Select start address and base address from **Start Address (Hex)** and **Base Address (Hex)** dropdown menus. For the AMD parallel flash, the starting address is automatically selected and cannot be changed.
8. If you want to erases the flash device if a verify error occurs, click, the **Erase Part on Program Error** box.
9. Click OK.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Configuring Advanced Flash Programming Options

You can use the Device Properties Dialog Box to configure Parallel Flash options.

To configure Parallel Flash options:

1. In Programmer, select the device that you want to edit.
2. Choose **Edit > Device Properties**, or click the button on the toolbar, or right-click and choose **Device Properties** to display the "Device Properties Dialog Box" on page 927.
3. In the Access Mode drop-down menu, choose **Advanced Flash Programming (FPGA Loader)**.
4. In the Advanced Flash Options box, select **Family**, **Vendor**, **Device**, **Package,** and **Operation**.
5. Click **Load Size from Programming File** to fill in the Data File Size box. You can change this size by typing a different file size. This feature is useful if you only want to use a portion of the file for the operation. If you change the file size, it must be no larger than the full file size or the device size.
6. Select start address and base address from **Start Address (Hex)** and **Base Address (Hex)** dropdown menus. For the AMD parallel flash, the starting address is automatically selected and cannot be changed.
7. If you want to erases the flash device if a verify error occurs, click, the **Erase Part on Program Error** box.
8. Click OK.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Configuring the BSCAN2

The "BSCAN Configuration Dialog Box" on page 942 enables you to configure five different BSCAN2 types. You can access up to 16 chains for programming by selecting multiple scan ports in these dialog boxes.

To configure the BSCAN2:

1. Choose **Design > BSCAN Configuration** to display the "BSCAN Configuration Dialog Box" on page 942.
2. In the BSCAN Configuration Type dropdown, choose from the following BSCAN configuration types:
   - BSCAN2 4 Port
   - BSCAN2 8 Port Asset Model
   - BSCAN2 8 Port JTAG/Corelis Model
   - BSCAN2 12 Port JTAG/Corelis Model

- BSCAN2 16 Port JTAG/Corelis Model.

Note

You can open the BSCAN2 Configuration dialog box even if you have no .xcf files open. You can also keep a BSCAN2 dialog box open and still have full access to the Programmer.

3. Select the multiple scan ports for the chains you want to access.
4. Accept the RTI default for Leave Selected Ports In.
5. To set all devices in the selected ports in the TLR state, choose **TLR** in the dropdown menu.

Note

Selecting Reset Selected Ports is an optional but recommended step. When you select this option, make sure you do so before selecting Configure Ports.

6. Click **Configure Ports.**

The software scans the ports and opens the chains and devices, including the BSCAN2 Linker, in the Programmer main window. A sync bit separates each device.

See Also

- "Using Diamond Programmer" on page 824

"Programming the FPGA" on page 802

Changing the Port Assignment

When you first launch the Programmer software with the cables properly connected, it connects to the first available port that it detects. You can change the connecting port, as well as other cable options, using the Cable and Port Setting dialog box.

To change the port setup:

- In the "Cable Settings Tab" on page 944, choose the options that you want. Changes are immediately applied.

  If the cable is not connected or cannot be detected (if the board power in not on, for example), Programmer software displays an error message.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Testing the Cable Signal

The Cable Signal Tests Dialog Box helps you debug or improve the cable signal. You can use an oscilloscope, while running the continuous loop feature, to test your printed circuit board.

To test the cable signal:

1. In the Cable Settings tab, click the **Debug Mode** button.

   The "Cable Signal Tests Dialog Box" on page 948 opens.

2. In the dialog box, click **Power Check** to test the connection.

   The VCC Status LED will blink green if the cable and power are detected. It will produce an error message and blink red if the cable or power is not detected.

3. In the dialog box, select an option for the pins you want to test. The options are defined as follows:

   - **Toggle**—Alternates between log 1, logic 0, and logic 1.
   - **Hold High**—Holds at logic 1.
   - **Hold Low**—Holds at logic 0.
   - **Read (TDO only)—**Reads back the data from TDO and records the read back data in the Programmer log file.

   Note

   The available options for cable signal testing will vary, depending on the type of cable you are using and the options you have selected in the Cable Settings tab. Options that are not available appear dimmed.

4. To test the settings you have selected, click **Test**.

   This causes the pins that have a Toggle setting to toggle once from logic 1 to logic 0 to logic 1.

5. To perform continuous testing, click **Loop Test**.

   This causes the pins that have a Toggle setting to toggle continuously until you stop the process.

6. To stop the loop process, press **ESC**.

See Also

- "Using Diamond Programmer" on page 824
- "Programming the FPGA" on page 802

Viewing the Log File

An ASCII text log file summarizes the results of the Programmer operations.

To view the log file:

- Click the  button on the toolbar.

See Also ▶"Using Diamond Programmer" on page 824

- "Programming the FPGA" on page 802

Programmer Options

This section lists the options available in Programmer.

Topics include:

- "Getting Started Dialog Box" on page 856
- "Device Properties Dialog Box—Device, Access Mode, and Operation Options" on page 857
- "Device Properties Dialog Box" on page 927
- "Edit Custom Device Dialog Box" on page 941
- "Custom SPI Flash Dialog Box" on page 942
- "BSCAN Configuration Dialog Box" on page 942
- "Settings Dialog Box" on page 943
- "Cable Settings Tab" on page 944
- "Firmware Tab" on page 946
- "Edit I/O State Dialog Box" on page 947
- "Display Status Register Dialog Box" on page 947
- ""Programmer Options" on page 856Custom Device Information Dialog Box" on page 947
- "Cable Signal Tests Dialog Box" on page 948

Getting Started Dialog Box

The following options are available in the Getting Started dialog box:

Create a New Project from a JTAG Scan Scans the attached chain with the last used cable settings.

Cable Specifies the download cable type:

- **HW-USBN-2A—**(Lattice HW-USBN-2A USB port programming cable)
- **HW-USBN-2B (FTDI)**—(Lattice HW-USBN-2B (FTDI) USB programming cable)

- **HW-DLN-3C (Parallel)**—(Lattice HW-DLN-3C parallel programming cable)

Port Specifies the serial port to which the download cable is connected. Select the port from the list or click Detect Cable.

Detect Cable Automatically detects the parallel port to which the download cable is connected.

Create a New Blank Project Creates a new blank project.

Open an Existing Programmer Project Allows you to browse to an existing .xcf configuration file and open an existing Programmer project.

See Also

- "Programmer Options" on page 856

Device Properties Dialog Box—Device, Access Mode, and Operation Options

The following tables list the Access Mode and Operation options available for the following Lattice devices. For detailed descriptions of Access Mode and Operation, refer to Device Properties Dialog Box.

Click the device name below to link to the Access Mode and Operation table for that device.

- ASC
- ispMACH4000
- ispMACH4000ZE
- ispPAC Platform Manager
- ispPAC Power Manager II
- ispPAC ProcessorPM
- ispXPGA
- ispXPLD 5000MX
- LatticeEC
- LatticeECP
- LatticeECP2
- LatticeECP2M
- LatticeECP2MS
- LatticeECP2S
- LatticeECP3

- LatticeECP3_AS
- LIFMD
- LIFMDF
- ECP5U
- ECP5UM
- LatticeSC
- LatticeSCM
- LatticeXP
- LatticeXP2
- LatticeXP2_AS
- MachXO
- MachXO2
- LFMNX and MachXO3D
- MachXO3LF
- ORCA FPSC
- ORCA4ExxE
- ORCA3LxxB
- ORCA2C/T
- ORCA3C/T

▶ Platform Manager 2

Table 33: ASC

Access Mode	Operation
I2C Interface Programming	I2C Erase, Program, Verify
	I2C Program
	I2C Erase Only
	I2C Verify Only
	I2C Verify ID
	I2C Read and Save
	I2C Read Status Register
	I2C Read FaultLog Registers
	I2C Read FaultLog Memory
	I2C FaultLog Erase Only
	I2C Erase,Program,Verify,I2CSA
	I2C I2CSA Program
	I2C I2CSA Erase Only
	I2C Background Erase, Program, Verify
	I2C Background Erase Only
	I2C Background Verify Only
	I2C Background Erase, Program, Verify, I2CSA

Return to Top

Table 34: ispMACH4000

Access Mode	Operation
JTAG 1532 Mode	Erase, Program, Verify
	Erase, Program, Verify, Secure
	Verify ID
	Display ID
	Verify Only
	Erase Only
	Bypass
	Calculate Checksum
	Verify USERCODE
	Display USERCODE
	Read and Save JEDEC
	Read DONE bit
	Erase DONE bit
	Program DONE bit

Return to Top

Table 35: ispMACH4000ZE

Access Mode	Operation
JTAG 1532 Mode	Erase, Program, Verify
	Erase, Program, Verify, Secure
	Verify ID
	Display ID
	Verify Only
	Erase Only
	Bypass
	Calculate Checksum
	Verify USERCODE
	Display USERCODE
	Read and Save JEDEC
	Read DONE bit
	Erase DONE bit
	Program DONE bit

Return to Top

Table 36: ispPAC Platform Manager

Access Mode	Operation
JTAG 1532 Mode	Erase, Program, Verify
	Erase, Program, Verify, Secure
	Verify ID
	Display ID
	Verify Only
	Erase Only
	Verify USERCODE
	Display USERCODE
	Calculate Checksum
	Read and Save JEDEC
	Secure Device
	Bypass

Return to Top

Table 37: ispPAC Power Manager II

Access Mode	Operation
JTAG 1532 Mode	Erase, Program, Verify
	Erase, Program, Verify, Secure
	Verify ID
	Display ID
	Verify Only
	Erase Only
	Bypass
	Display USERCODE
	Calculate Checksum
	Read and Save JEDEC
	Secure Device
ATDI Pin Selected Programming	ATDI Erase, Program, Verify
	ATDI Erase, Program, Verify, Secure
	ATDI Verify ID
	ATDI Display ID
	ATDI Verify Only
	ATDI Erase Only
	ATDI Bypass
	ATDI Display USERCODE
	ATDI Secure Device

Return to Top

Table 38: ispPAC ProcessorPM

Access Mode	Operation
JTAG 1532 Mode	Erase, Program, Verify
	Erase, Program, Verify, Secure
	Verify ID
	Display ID
	Verify Only
	Erase Only
	Read and Save JEDEC
	Bypass
	Display USERCODE
	Secure Device
ATDI Pin Selected Programming	ATDI Erase, Program, Verify
	ATDI Erase, Program, Verify, Secure
	ATDI Verify ID
	ATDI Display ID
	ATDI Verify Only
	ATDI Erase Only
	ATDI Bypass
	ATDI Display USERCODE
	ATDI Secure Device

Return to Top

Table 39: ispXPGA

Access Mode	Operation
EE	Erase, Program, Verify
	Erase, Program, Verify, Secure
	Verify ID
	Display ID
	Verify Only
	Erase Only
	Bypass
	Calculate Data CRC
	Verify USERCODE
	Display USERCODE
	Read and Save
	Read DONE bit
	Erase DONE bit
	Program DONE bit
	Refresh
X-EE	XEE Erase, Program, Verify
	XEE Erase, Program, Verify, Refresh
	XEE Erase, Program, Verify, Secure
	XEE Verify ID
	XEE Display ID
	XEE Verify Only
	XEE Erase Only
	XEE Bypass
	XEE Calculate Data CRC
	XEE Verify USERCODE
	XEE Display USERCODE
	XEE Read and Save
	XEE Read DONE bit
	XEE Erase DONE bit
	XEE Program DONE bit

Table 39: ispXPGA (Continued)

Access Mode	Operation
Static RAM Cell Mode	SRAM Program, Verify
	SRAM Program Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Erase Only
	SRAM Bypass
	SRAM Calculate Data CRC
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read DONE bit
	SRAM Erase DONE bit
	SRAM Program DONE bit
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Bypass
	XSRAM Calculate Data CRC
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read DONE bit

Return to Top

Table 40: ispXPLD 5000MX

Access Mode	Operation
EE	Erase, Program, Verify
	Erase, Program, Verify, Secure
	Verify ID
	Display ID
	Verify Only
	Erase Only
	Bypass
	Calculate Checksum
	Verify USERCODE
	Display USERCODE
	Read and Save
	Read DONE bit
	Erase DONE bit
	Program DONE bit
	Refresh

Table 40: ispXPLD 5000MX (Continued)

Access Mode	Operation
X-EE	XEE Erase, Program, Verify
	XEE Erase, Program, Verify, Refresh
	XEE Erase, Program, Verify, Secure
	XEE Verify ID
	XEE Display ID
	XEE Verify Only
	XEE Erase Only
	XEE Bypass
	XEE Calculate Checksum
	XEE Verify USERCODE
	XEE Display USERCODE
	XEE Read and Save
	XEE Read DONE bit
	XEE Erase DONE bit
	XEE Program DONE bit
Static RAM Cell Mode	SRAM Program, Verify
	SRAM Program Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Erase Only
	SRAM Bypass
	SRAM Calculate Checksum
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read DONE bit
	SRAM Erase DONE bit
	SRAM Program DONE bit

Return to Top

Table 41: LatticeEC

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Programming Pins Status
	Display Control Register0
	Refresh
	Bypass
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Read and Save
	XSRAM Bypass
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 42: LatticeECP

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Programming Pins Status
	Display Control Register0
	Refresh
	Bypass
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Read and Save
	XSRAM Bypass
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 43: LatticeECP2

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Status Register
	Refresh from FLASH
	FLASH TransFR
	Verify DONE bit
	Bypass
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
	SPI Flash Background Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify, Refresh
	SPI Flash Background Program
	SPI Flash Background Verify Only
	SPI Flash Background Erase All
	SPI Flash Background Read and Save
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Read and Save
	XSRAM Verify DONE bit
	XSRAM Bypass
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 44: LatticeECP2M

Access Mode	Operation
JTAG 1532	Fast Program
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Status Register
	Refresh from FLASH
	FLASH TransFR
	Verify DONE bit
	Bypass
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
	SPI Flash Background Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify, Refresh
	SPI Flash Background Erase, Program
	SPI Flash Background Verify Only
	SPI Flash Background Erase All
	SPI Flash Background Read and Save
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Read and Save
	XSRAM Verify DONE bit
	XSRAM Bypass
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 45: LatticeECP2MS

Access Mode	Operation
JTAG 1532	Fast Program
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Status Register
	Refresh from FLASH
	FLASH TransFR
	Verify DONE bit
	Bypass
Advanced Security Keys Programming	Security Program Encryption Key
	Security Read Encryption Key
	Security Program Key Lock
Advanced Security Production Programming	Security Fast Program with Encryption Option
SPI Flash Programming	SPI Flash Erase ,Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
	SPI Flash Background Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify, Refresh
	SPI Flash Background Erase, Program
	SPI Flash Background Verify Only
	SPI Flash Background Erase All
	SPI Flash Background Read and Save

Table 45: LatticeECP2MS (Continued)

Access Mode	Operation
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Read and Save
	XSRAM Verify DONE bit
	XSRAM Bypass
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 46: LatticeECP2S

Access Mode	Operation
JTAG 1532	Fast Program
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Status Register
	Refresh from FLASH
	FLASH TransFR
	Verify DONE bit
	Bypass
Advanced Security Keys Programming	Security Program Encryption Key
	Security Read Encryption Key
	Security Program Key Lock
Advanced Security Production Programming	Security Fast Program with Encryption Option

Table 46: LatticeECP2S (Continued)

Access Mode	Operation
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
	SPI Flash Background Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify, Refresh
	SPI Flash Background Erase, Program
	SPI Flash Background Verify Only
	SPI Flash Background Erase All
	SPI Flash Background Read and Save
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Read and Save
	XSRAM Verify DONE bit
	XSRAM Bypass
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 47: LatticeECP3

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Erase, Program, Verify
	Verify Only
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Read DONE bit
	Display Status Register
	Display Control Register0
	Refresh from FLASH
	Bypass
Advanced Security Keys Programming	Security Program Encryption Key
	Security Verify Encryption Key
	Security Read Encryption Key
	Security Program Key Lock
Advanced Security Production Programming	Security Read Status Register
	Security Fast Program with Encryption Option
	Security Verify with Encryption Option
Slave SPI Interface Programming	Slave SPI Fast Program
	Slave SPI Fast Program, Refresh
	Slave SPI Erase Only
	Slave SPI Verify ID
	Slave SPI Display ID
	Slave SPI Display USERCODE
	Slave SPI Read and Save
	Slave SPI Display Control Register0
	Slave SPI Read Status Register
	Slave SPI Upload to Static RAM

Table 47: LatticeECP3 (Continued)

Access Mode	Operation
SPI Flash Background Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
Dual Boot SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Read and Save
	XSRAM Display Status Register
	XSRAM Bypass
Serial Mode	Serial Program
	Serial Read DONE bit
SSPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device

Return to Top

Table 48: LatticeECP3_AS

Access Mode	Operation
JTAG 1532 Mode	Erase, Program, Verify
	Verify Only
	Erase Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Read DONE bit
	Display Status Register
	Display Control Register0
	Bypass

Return to Top

Table 49: LIFMD

Access Mode	Operation
SSPI SRAM Programming	Fast Program
	Erase, Program, Verify
	Erase Only
	Verify ID
	Display ID
	Read Status Register
SSPI XSRAM Programming	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
SPI Flash Background Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Bypass
	Scan SPI Flash Device

Table 49: LIFMD (Continued)

Access Mode	Operation
I2C Interface Programming	I2C Fast Program
	I2C Erase, Program, Verify
	I2C Erase Only
	I2C Verify ID
	I2C Display ID
	I2C Read Status Register
SSPI NVCM Programming	SSPI NVCM Program, Verify
	SSPI NVCM Program, Verify, Secure
	SSPI NVCM Blank Check
	SSPI NVCM Verify Only
	SSPI NVCM Verify ID
	SSPI NVCM Display ID
	SSPI NVCM Read Status Register
I2C NVCM Programming	I2C NVCM Program, Verify
	I2C NVCM Program, Verify, Secure
	I2C NVCM Blank Check
	I2C NVCM Verify Only
	I2C NVCM Verify ID
	I2C NVCM Display ID
	I2C NVCM Read Status Register

Return to Top

Table 50: LIFMDF

Access Mode	Operation
Static RAM Cell Mode	SRAM Fast Configuration
	SRAM Erase, Program, Verify
	SRAM Verify Only
	SRAM Erase Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Program Control Register0
	SRAM Display Control Register0
	SRAM Program Control Register1
	SRAM Display Control Register1
	SRAM Verify USERCODE
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read Status Register
	SRAM Refresh
	SRAM Bypass
Static RAM Cell Background Mode	XSRAM SEI Fast Program
	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Display Control Register0
	XSRAM Display Control Register1
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
	XSRAM Bypass
SSPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Erase, Program, Verify Quad 1
	SPI Flash Verify ID
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Bypass
	Scan SPI Flash Device

Table 50: LIFMDF (Continued)

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Verify Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH CFG Erase, Program, Verify
	FLASH CFG Verify Only
	FLASH CFG Erase Only
	FLASH UFM Erase, Program, Verify
	FLASH UFM Verify Only
	FLASH UFM Erase Only
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH Read Status Register
	FLASH Refresh
	FLASH Bypass
Flash Background Mode	XFLASH Erase, Program, Verify
	XFLASH Verify Only
	XFLASH Erase Only
	XFLASH Verify ID
	XFLASH Display ID
	XFLASH Program and TransFR
	XFLASH TransFR
	XFLASH CFG Erase, Program, Verify
	XFLASH CFG Verify Only
	XFLASH CFG Erase Only
	XFLASH UFM Erase, Program, Verify
	XFLASH UFM Verify Only
	XFLASH UFM Erase Only
	XFLASH Display USERCODE
	XFLASH Read and Save
	XFLASH Calculate Checksum
	XFLASH Read Status Register
	XFLASH Refresh
	XFLASH Bypass

Table 50: LIFMDF (Continued)

Access Mode	Operation
Feature Row Programming	Program Feature Row
	Update Feature Row
	Erase Feature Row
	Read Feature Row
	Program Control NV Register1
	Display Control NV Register1
Advanced Security Keys Programming	Security Program Password Key
	Security Read Password Key
	Security Program TraceID
	Security Read TraceID

Return to Top

Table 51: ECP5U

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Erase, Program, Verify
	Erase Only
	Verify Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Control Register0
	Program Control Register0
	Read Status Register
	FLASH TransFR
	Refresh
	Bypass
Static RAM Cell Background Mode	XSRAM SEI Fast Program
	XSRAM Verify Only
	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Verify USERCODE
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
Advanced Security Keys Programming	Security Program Encryption Key Only
	Security Program Feature Lock
	Security Read Encryption Key
	Security Program TraceID
	Security Read TraceID
	Security Read Status Register
	Manufacturing Read DTR Fuses
	Manufacturing Read DTR Fuses Only
	Read Manufacturing Register
	Program Manufacturing Register

Table 51: ECP5U (Continued)

Access Mode	Operation
SPI Flash Background Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
Slave SPI Interface Programming	Slave SPI Fast Program
	Slave SPI SEI Fast Program
	Slave SPI Erase Only
	Slave SPI Verify ID
	Slave SPI Read and Save
	Slave SPI Read Status Register
	Slave SPI Upload to Static RAM
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 52: ECP5UM

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Erase, Program, Verify
	Erase Only
	Verify Only
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Display Control Register0
	Program Control Register0
	Read Status Register
	FLASH TransFR
	Refresh
	Bypass
Static RAM Cell Background Mode	XSRAM SEI Fast Program
	XSRAM Verify Only
	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Verify USERCODE
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
	XSRAM Bypass
Advanced Security Keys Programming	Security Program Encryption Key Only
	Security Program Feature Lock
	Security Read Encryption Key
	Security Program TraceID
	Security Read TraceID
	Security Read Status Register
	Manufacturing Read DTR Fuses
	Manufacturing Read DTR Fuses Only
	Read Manufacturing Register
	Program Manufacturing Register

Table 52: ECP5UM (Continued)

Access Mode	Operation
SPI Flash Background Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
Slave SPI Interface Programming	Slave SPI Fast Program
	Slave SPI SEI Fast Program
	Slave SPI Erase Only
	Slave SPI Verify ID
	Slave SPI Read and Save
	Slave SPI Read Status Register
	Slave SPI Upload to Static RAM
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 53: LatticeSC

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Read DONE bit
	Bypass
	Refresh from FLASH
	Read Programming Status

Table 53: LatticeSC (Continued)

Access Mode	Operation
Serial Mode	Serial Program
	Serial Read DONE bit
SPI Flash Background Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass

Return to Top

Table 54: LatticeSCM

Access Mode	Operation
JTAG 1532 Mode	Fast Program
	Verify ID
	Display ID
	Display USERCODE
	Read and Save
	Read DONE bit
	Bypass
	Refresh from FLASH
	Read Programming Status
Serial Mode	Serial Program
	Serial Read DONE bit
SPI Flash Background Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass

Return to Top

Table 55: LatticeXP

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify, Secure
	FLASH Program
	FLASH Verify Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH Read DONE bit
	FLASH Bypass
Flash Background Mode	XFLASH Erase, Program, Verify
	XFLASH Erase, Program, Verify, Secure
	XFLASH Program
	XFLASH Program and TransFR
	XFLASH TransFR
	XFLASH Verify Only
	XFLASH Erase Only
	XFLASH Display USERCODE
	XFLASH Read and Save
	XFLASH Calculate Checksum
	XFLASH Upload to Static RAM
	XFLASH Bypass
Static RAM Cell Mode	SRAM Program, Verify
	SRAM Verify ID
	SRAM Display ID
	SRAM Verify Only
	SRAM Erase Only
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read DONE bit
	SRAM Refresh
	SRAM Bypass

Table 55: LatticeXP (Continued)

Access Mode	Operation
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Bypass
Serial Mode	Serial Program

Return to Top

Table 56: LatticeXP2

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Erase, Program, Verify, Secure
	FLASH Erase, Program, Verify, Refresh
	FLASH Program
	FLASH Verify Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH TAG Program, Verify
	FLASH TAG Erase
	FLASH Save JEDEC with TAG
	FLASH Read Status Register
	FLASH Verify DONE bit
	FLASH Read DONE bit
	FLASH Refresh
	FLASH Bypass

Table 56: LatticeXP2 (Continued)

Access Mode	Operation
Flash Background Mode	XFLASH Erase, Program, Verify
	XFLASH Erase, Program, Verify, Secure
	XFLASH Erase, Program, Secure
	XFLASH Program
	XFLASH Program and TransFR
	XFLASH TransFR
	XFLASH Verify Only
	XFLASH Erase Only
	XFLASH Display USERCODE
	XFLASH Read and Save
	XFLASH Calculate Checksum
	XFLASH Upload to Static RAM
	XFLASH TAG Program, Verify
	XFLASH TAG Erase
	XFLASH Save JEDEC with TAG
	XFLASH Read Status Register
	XFLASH Verify DONE bit
	XFLASH Read DONE bit
	XFLASH Refresh
	XFLASH Bypass
Static RAM Cell Mode	SRAM Program, Verify
	SRAM Verify ID
	SRAM Display ID
	SRAM Erase Only
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read Status Register
	SRAM Read DONE bit
	SRAM Refresh
	SRAM Bypass
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Verify Only
	XSRAM Read Status Register
	XSRAM Read and Save
	XSRAM Bypass

Table 56: LatticeXP2 (Continued)

Access Mode	Operation
Advanced Security Keys Programming	Security Program Encryption Key Only
	Security Program Encryption Key with Lock
	Security Program Flash Protect Key Only
	Security Program Flash Protect Key with Lock
	Security Program Encryption Key and Flash Protect
	Security Program Encryption Key and Flash Protect with Lock
	Security Read Feature Row
	Security Read Status Register
	Security Erase Feature Row
Advanced Security Encryption File Programming	Security Read Status Register
	Security EPVS with Encrypt Only Option
	Security EPVS with Encrypt and Lock Option
	Security EPV with Protect Only Option
	Security EPV with Protect and Lock Option
	Security EPVS with Protect Only Option
	Security EPVS with Protect and Lock Option
	Security EPVS with full feature Option
	Security EPVS with full feature and Lock Option
	Security EPV with OTP Option
	Security EPVS with OTP Option
	Security EPV with my_ASSP Option
	Security EPVS with my_ASSP Option
	Security EPV with OTP and my_ASSP Option
	Security EPVS with OTP and my_ASSP Option
	Security Erase Feature Row

Table 56: LatticeXP2 (Continued)

Access Mode	Operation
Advanced Security Production Programming	Security Read Status Register
	Security EPVS with Encrypt Only Option
	Security EPVS with Encrypt and Lock Option
	Security EPV with Protect Only Option
	Security EPV with Protect and Lock Option
	Security EPVS with Protect Only Option
	Security EPVS with Protect and Lock Option
	Security EPVS with full feature Option
	Security EPVS with full feature and Lock Option
	Security EPV with OTP Option
	Security EPVS with OTP Option
	Security EPV with my_ASSP Option
	Security EPVS with my_ASSP Option
	Security EPV with OTP and my_ASSP Option
	Security EPVS with OTP and my_ASSP Option
	Security Erase Feature Row

Table 56: LatticeXP2 (Continued)

Access Mode	Operation
Slave SPI Interface Programming	Slave SPI Erase, Program, Verify
	Slave SPI Erase, Program, Verify, Secure
	Slave SPI Erase, Program, Verify, Secure, Refresh
	Slave SPI Erase, Program, Verify, Refresh
	Slave SPI Background Erase, Program
	Slave SPI Background Erase, Program, Secure
	Slave SPI Background Erase, Program, Secure, Refresh
	Slave SPI Verify Only
	Slave SPI Erase Only
	Slave SPI Background Erase Only
	Slave SPI Verify ID
	Slave SPI Display ID
	Slave SPI Display USERCODE
	Slave SPI Read and Save
	Slave SPI Read Status Register
	Slave SPI RAM Verify Only
	Slave SPI RAM Read and Save
	Slave SPI TAG Program, Verify
	Slave SPI TAG Verify
	Slave SPI TAG Erase
	Slave SPI TAG Read and Program
	Slave SPI Save JEDEC with TAG
	Slave SPI Upload to Static RAM
	Slave SPI EPVS with Encryption
	Slave SPI EPVS with Encryption, Refresh
	Slave SPI EPV with Flash Protect
	Slave SPI EPV with Flash Protect, Refresh
	Slave SPI EPVS with Flash Protect
	Slave SPI EPS with Flash Protect
	Slave SPI EPVS with Flash Protect, Refresh
	Slave SPI EPS with Flash Protect, Refresh
	Slave SPI EPVS with Encryption and Flash Protect
	Slave SPI EPVS with Encryption and Flash Protect, Refresh
	Slave SPI Erase with Flash Protect
	Slave SPI Refresh

Table 56: LatticeXP2 (Continued)

Access Mode	Operation
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Erase, Program
	SPI Flash Background Erase, Program
	SPI Flash Verify Only
	SPI Flash Background Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device

Return to Top

Table 57: LatticeXP2_AS

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Erase, Program, Verify, Secure
	FLASH Verify Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH TAG Program, Verify
	FLASH TAG Erase
	FLASH Save JEDEC with TAG
	FLASH Read Status Register
	FLASH Read DONE bit
	FLASH Bypass
Flash Background Mode	XFLASH Erase, Program, Verify
	XFLASH Erase, Program, Verify, Secure
	XFLASH Program and TransFR
	XFLASH TransFR
	XFLASH Verify Only
	XFLASH Erase Only
	XFLASH Read and Save
	XFLASH Upload to Static RAM
	XFLASH TAG Program, Verify
	XFLASH TAG Erase
	XFLASH TAG Read and Program
	XFLASH Save JEDEC with TAG
	XFLASH Read Status Register
	XFLASH Bypass

Table 57: LatticeXP2_AS (Continued)

Access Mode	Operation
Static RAM Cell Mode	SRAM Program, Verify
	SRAM Verify ID
	SRAM Display ID
	SRAM Erase Only
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read Status Register
	SRAM Read DONE bit
	SRAM Refresh
	SRAM Bypass
Static RAM Cell Background Mode	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Verify Only
	XSRAM Read Status Register
	XSRAM Read and Save
	XSRAM Bypass
Advanced Security Encryption File Programming	Security Erase, Program, Verify
	Security Erase Only
	Security Erase, Program, Verify with my_ASSP
	Security Erase, Program, Verify with OTP
	Security Background Erase, Program, Verify
	Security Background Program and TransFR
	Security Background Erase Only
	Security Read Status Register
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Erase, Program, Verify, Refresh
	SPI Flash Erase, Program
	SPI Flash Verify Only
	SPI Flash Erase All
	SPI Flash Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass

Return to Top

Table 58: MachXO

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Erase, Program, Verify, Secure
	FLASH Verify Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH Program DONE bit
	FLASH Read DONE bit
	FLASH Secure Device
	FLASH Bypass
Flash Background Mode	XFLASH Erase, Program, Verify
	XFLASH Erase, Program, Verify, Secure
	XFLASH Program and TransFR
	XFLASH Program, Secure and TransFR
	XFLASH TransFR
	XFLASH Program USF from PROM file
	XFLASH Verify Only
	XFLASH Erase Only
	XFLASH Read and Save
	XFLASH Calculate Checksum
	XFLASH Display USERCODE
	XFLASH Upload to Static RAM
	XFLASH Secure Device
	XFLASH Bypass

Table 58: MachXO (Continued)

Access Mode	Operation
Static RAM Cell Mode	SRAM Program, Verify
	SRAM Program Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Erase Only
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read DONE bit
	SRAM Secure Device
	SRAM Bypass
Static RAM Cell Background Mode	XSRAM Display ID
	XSRAM Read and Save
	XSRAM Display USERCODE
	XSRAM Bypass

Return to Top

Table 59: MachXO2

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Erase, Program, Verify, Secure
	FLASH Erase, Program, Verify, Secure Plus
	FLASH Program
	FLASH CFG Erase, Program, Verify
	FLASH CFG and UFM Erase, Program, Verify
	FLASH Verify Only
	FLASH Erase Only
	FLASH Erase CFG Only
	FLASH Erase CFG and UFM Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH UFM Program, Verify
	FLASH UFM Erase
	FLASH Verify UFM
	FLASH Read Status Register
	FLASH Secure Device
	FLASH Secure Plus
	FLASH Refresh
	FLASH Verify Feature Rows
	FLASH Program Feature Rows
	FLASH Bypass

Table 59: MachXO2 (Continued)

Access Mode	Operation
Flash Background Mode	XFLASH Erase, Program, Verify
	XFLASH Erase, Program, Verify, Secure
	XFLASH Erase, Program, Verify, Secure Plus
	XFLASH Erase, Program, Verify, Refresh
	XFLASH Verify Feature Rows
	XFLASH Program Feature Rows
	XFLASH Erase, Program, Verify, Feature
	XFLASH CFG Erase, Program, Verify
	XFLASH Erase, Program, Verify, Feature, and TransFR
	XFLASH Program and TransFR
	XFLASH TransFR
	XFLASH Erase Only
	XFLASH Erase CFG Only
	XFLASH Verify Only
	XFLASH Read and Save
	XFLASH Calculate Checksum
	XFLASH UFM Program, Verify
	XFLASH UFM Erase
	XFLASH Read Status Register
	XFLASH Secure Device
	XFLASH Secure Plus
	XFLASH Bypass
Static RAM Cell Mode	SRAM Fast Program
	SRAM Erase, Program, Verify
	SRAM Erase Only
	SRAM Verify Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Display Control Register0
	SRAM Program Control Register0
	SRAM Read Status Register
	SRAM Refresh
	SRAM Bypass

Table 59: MachXO2 (Continued)

Access Mode	Operation
Static RAM Cell Background Mode	XSRAM SEI Fast Program
	XSRAM Verify Only
	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Verify USERCODE
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
	XSRAM Bypass
Advanced Security Keys Programming	Security Read Feature Rows
	Security Program Feature Rows
	Security Program Password Key
	Security Program Password Key with Lock
	Security Erase Feature Row with Password Key
	Security Read Status Register
	Security Display TraceID
Advanced Security File Programming	Security Read Status Register
	Security Display TraceID
	Security Flash EPV with Password
	Security Flash Program with Password
	Security Flash Verify with Password
	Security Flash Erase with Password
	Security XFlash EPV with Password
	Security XFlash Program with Password
	Security XFlash Verify with Password
	Security XFlash Erase with Password
	Security Fast Program with Password
	Security SRAM EPV with Password
	Security SRAM Verify with Password
	Security SRAM Erase with Password
	Security XSRAM SEI Fast Program with Password
	Security XSRAM Verify with Password

Table 59: MachXO2 (Continued)

Access Mode	Operation
Advanced Security Production Programming	Security Read Status Register
	Security EPV with Password Key Option
	Security EPV with my_ASSP, Password Key Option
	Security EPV with my_ASSP Option
	Security EPVS with my_ASSP Option
	Security EPV with OTP Option
	Security EPVS with OTP Option
	Security EPV with Full OTP Option
	Security EPVS with Full OTP Option
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Background Erase, Program
	SPI Flash Verify Only
	SPI Flash Background Verify Only
	SPI Flash Erase All
	SPI Flash Background Erase All
	SPI Flash Read and Save
	SPI Flash Background Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device

Table 59: MachXO2 (Continued)

Access Mode	Operation
Slave SPI Interface Programming	Slave SPI Erase, Program, Verify
	Slave SPI Fast Configuration
	Slave SPI Erase, Program, Verify, Secure
	Slave SPI Erase, Program, Verify, Secure Plus
	Slave SPI CFG Erase, Program, Verify
	Slave SPI Erase Only
	Slave SPI Verify Only
	Slave SPI Verify ID
	Slave SPI Display ID
	Slave SPI Display USERCODE
	Slave SPI Read and Save
	Slave SPI Read Status Register
	Slave SPI UFM Program, Verify
	Slave SPI UFM Erase
	Slave SPI Background Erase, Program, Verify
	Slave SPI Background Erase, Program, Verify, Secure
	Slave SPI Background Erase, Program, Verify, Secure Plus
	Slave SPI Background CFG Erase, Program, Verify
	Slave SPI Background Program Feature Rows
	Slave SPI Background Erase, Program, Verify, Feature
	Slave SPI Background Erase Only
	Slave SPI Background Verify Only
	Slave SPI Background SEI Fast Program
	Slave SPI Refresh
	Slave SPI Erase, Program, Verify with Password
	Slave SPI Fast Configuration with Password
	Slave SPI Verify Only with Password
	Slave SPI Erase Only with Password

Table 59: MachXO2 (Continued)

Access Mode	Operation
I2C Interface Programming	I2C Erase, Program, Verify
	I2C Program
	I2C Erase, Program, Verify, Secure
	I2C Erase, Program, Verify, Secure Plus
	I2C Erase Only
	I2C Verify Only
	I2C Verify ID
	I2C Display ID
	I2C Program Feature Rows
	I2C Erase, Program, Verify, Feature
	I2C Recovery Erase Only
	I2C Background Erase, Program, Verify
	I2C Background Erase, Program, Verify, Secure
	I2C Background Erase, Program, Verify, Secure Plus
	I2C Background Program
	I2C Background Erase Only
	I2C Background Verify Only
	I2C Background Program Feature Rows
	I2C Refresh
	I2C Program Password Key
	I2C Program Password Key with Lock
	I2C Erase Feature Row with Password Key
	I2C EPV with Password Key Option
	I2C EPV with my_ASSP, Password Key Option
	I2C Erase, Program, Verify with Password
	I2C Fast Configuration with Password
	I2C Verify Only with Password
	I2C Erase Only with Password

Return to Top

Table 60: LFMNX and MachXO3D

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Verify Only
	FLASH CFG Erase, Program, Verify
	FLASH CFG Verify Only
	FLASH CFG Erase Only
	FLASH UFM Erase, Program, Verify
	FLASH UFM Verify Only
	FLASH UFM Erase Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH Read Status Register
	FLASH Refresh
	FLASH Dry Run
	FLASH Version Rollback Protection
	FLASH Bypass
Flash Background Mode	FLASH Erase, Program, Verify
	FLASH Verify Only
	FLASH CFG Erase, Program, Verify
	FLASH CFG Verify Only
	FLASH CFG Erase Only
	FLASH UFM Erase, Program, Verify
	FLASH UFM Verify Only
	FLASH UFM Erase Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH Read Status Register
	FLASH Refresh
	FLASH Dry Run
	FLASH Bypass

Table 60: LFMNX and MachXO3D (Continued)

Access Mode	Operation
Static RAM Cell Mode	SRAM Fast Configuration
	SRAM Erase, Program, Verify
	SRAM Verify Only
	SRAM Erase Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Program Control Register0
	SRAM Display Control Register0
	SRAM Program Control Register1
	SRAM Display Control Register1
	SRAM Verify USERCODE
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Read Status Register
	SRAM Refresh
	SRAM Bypass
	SRAM External Primary Dry Run
	SRAM External Golden Dry Run
Static RAM Cell Background Mode	XSRAM SEI Fast Program
	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Display Control Register0
	XSRAM Display Control Register1
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
	XSRAM Bypass

Table 60: LFMNX and MachXO3D (Continued)

Access Mode	Operation
Advanced Security Keys Programming	Security Program Password Key
	Security Read Password Key
	Security Program Encryption Key
	Security Read Encryption Key
	Security Erase Encryption Key
	Security Program Public Key
	Security Read Public Key
	Security Erase Public Key
	Security Program TraceID
	Security Read TraceID
	Security Program Auth Done FlashA
	Security Program Auth Done FlashB
Feature Row Programming	Program Feature Row
	Update Feature Row
	Erase Feature Row
	Read Feature Row
	Program Control NV Register1
	Display Control NV Register1
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Verify Only
	SPI Flash Erase All
	SSPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Read and Save
	SPI Flash Erase, Program, Verify Quad 1
	Scan SPI Flash Device

Return to Top

Table 61: MachXO3L

Access Mode	Operation
NVCM Programming Mode	NVCM Erase, Program, Verify
	NVCM Erase, Program, Verify, Secure
	NVCM Program
	NVCM Verify Only
	NVCM Erase Only
	NVCM Verify ID
	NVCM Display ID
	NVCM Display USERCODE
	NVCM Read and Save
	NVCM Calculate Checksum
	NVCM Read Status Register
	NVCM Secure Device
	NVCM Refresh
	NVCM Bypass
Static RAM Cell Mode	SRAM Fast Configuration
	SRAM Erase, Program, Verify
	SRAM Erase Only
	SRAM Verify Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Display Control Register0
	SRAM Program Control Register0
	SRAM Read Status Register
	SRAM Refresh
	SRAM Bypass

Table 61: MachXO3L (Continued)

Access Mode	Operation
Static RAM Cell Background Mode	XSRAM SEI Fast Program
	XSRAM Verify Only
	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Verify USERCODE
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
	XSRAM Bypass
Advanced Security Keys Programming	Security Read Feature Rows
	Security Program Feature Rows
	Security Program Password Key
	Security Program Password Key with Lock
	Security Erase Feature Row with Password Key
Advanced Security File Programming	Security Read Status Register
	Security NVCM EPV with Password
	Security NVCM Program with Password
	Security NVCM Verify with Password
	Security NVCM Erase with Password
	Security Fast Program with Password
	Security SRAM EPV with Password
	Security SRAM Verify with Password
	Security SRAM Erase with Password
	Security XSRAM SEI Fast Program with Password
	Security XSRAM Verify with Password
Advanced Security Production Programming	Security Read Status Register
	Security EPV with Password Key Option
	Security EPV with my_ASSP, Password Key Option
	Security EPV with OTP Option
	Security EPVS with OTP Option
	Security EPV with Full OTP Option
	Security EPVS with Full OTP Option

Table 61: MachXO3L (Continued)

Access Mode	Operation
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Background Erase, Program
	SPI Flash Verify Only
	SPI Flash Background Verify Only
	SPI Flash Erase All
	SPI Flash Background Erase All
	SPI Flash Read and Save
	SPI Flash Background Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device

Table 61: MachXO3L (Continued)

Access Mode	Operation
Slave SPI Interface Programming	Slave SPI Erase, Program, Verify
	Slave SPI Fast Configuration
	Slave SPI Erase, Program, Verify, Secure
	Slave SPI Erase Only
	Slave SPI Verify Only
	Slave SPI Verify ID
	Slave SPI Display ID
	Slave SPI Display USERCODE
	Slave SPI Read and Save
	Slave SPI Read Status Register
	Slave SPI Background SEI Fast Program
	Slave SPI Refresh
	Slave SPI Erase, Program, Verify with Password
	Slave SPI Fast Configuration with Password
	Slave SPI Verify Only with Password
	Slave SPI Erase Only with Password
I2C Interface Programming	I2C Erase, Program, Verify
	I2C Erase, Program, Verify, Feature
	I2C Fast Configuration
	I2C Program
	I2C Erase, Program, Verify, Secure
	I2C Erase Only
	I2C Verify Only
	I2C Verify ID
	I2C Display ID
	I2C Program Feature Rows
	I2C Recovery Erase Only
	I2C Refresh
	I2C Program Password Key
	I2C Program Password Key with Lock
	I2C Erase Feature Row with Password Key
	I2C EPV with Password Key Option
	I2C EPV with my_ASSP, Password Key Option
	I2C Erase, Program, Verify with Password
	I2C Fast Configuration with Password
	I2C Verify Only with Password
	I2C Erase Only with Password

Return to Top

Table 62: MachXO3LF

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Erase, Program, Verify, Secure
	FLASH Erase, Program, Verify, Secure Plus
	FLASH Program
	FLASH CFG Erase, Program, Verify
	FLASH CFG and UFM Erase, Program, Verify
	FLASH Verify Only
	FLASH Erase Only
	FLASH Erase CFG Only
	FLASH Erase CFG and UFM Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH UFM Program, Verify
	FLASH UFM Erase
	FLASH Verify UFM
	FLASH Read Status Register
	FLASH Secure Device
	FLASH Secure Plus
	FLASH Refresh
	FLASH Verify Feature Rows
	FLASH Program Feature Rows
	FLASH Bypass

Table 62: MachXO3LF (Continued)

Access Mode	Operation
Flash Background Mode	XFLASH Erase, Program, Verify
	XFLASH Erase, Program, Verify, Secure
	XFLASH Erase, Program, Verify, Secure Plus
	XFLASH Erase, Program, Verify, Refresh
	XFLASH Verify Feature Rows
	XFLASH Program Feature Rows
	XFLASH Erase, Program, Verify, Feature
	XFLASH Erase, Program, Verify, Feature, and TransFR
	XFLASH CFG Erase, Program, Verify
	XFLASH Program and TransFR
	XFLASH TransFR
	XFLASH Erase Only
	XFLASH Erase CFG Only
	XFLASH Verify Only
	XFLASH Read and Save
	XFLASH Calculate Checksum
	XFLASH UFM Program, Verify
	XFLASH UFM Erase
	XFLASH Read Status Register
	XFLASH Secure Device
	XFLASH Secure Plus
	XFLASH Bypass
Static RAM Cell Mode	SRAM Fast Program
	SRAM Erase, Program, Verify
	SRAM Erase Only
	SRAM Verify Only
	SRAM Verify ID
	SRAM Display ID
	SRAM Display USERCODE
	SRAM Read and Save
	SRAM Display Control Register0
	SRAM Program Control Register0
	SRAM Read Status Register
	SRAM Refresh
	SRAM Bypass

Table 62: MachXO3LF (Continued)

Access Mode	Operation
Static RAM Cell Background Mode	XSRAM SEI Fast Program
	XSRAM Verify Only
	XSRAM Verify ID
	XSRAM Display ID
	XSRAM Verify USERCODE
	XSRAM Display USERCODE
	XSRAM Read and Save
	XSRAM Read Status Register
	XSRAM Refresh
	XSRAM Bypass
Advanced Security Keys Programming	Security Read Feature Rows
	Security Program Feature Rows
	Security Program Password Key
	Security Program Password Key with Lock
	Security Erase Feature Row with Password Key
	Security Read Status Register
	Security Display TraceID
Advanced Security File Programming	Security Read Status Register
	Security Display TraceID
	Security Flash EPV with Password
	Security Flash Program with Password
	Security Flash Verify with Password
	Security Flash Erase with Password
	Security XFlash EPV with Password
	Security XFlash Program with Password
	Security XFlash Verify with Password
	Security XFlash Erase with Password
	Security Fast Program with Password
	Security SRAM EPV with Password
	Security SRAM Verify with Password
	Security SRAM Erase with Password
	Security XSRAM SEI Fast Program with Password
	Security XSRAM Verify with Password

Table 62: MachXO3LF (Continued)

Access Mode	Operation
Advanced Security Production Programming	Security Read Status Register
	Security EPV with Password Key Option
	Security EPV with my_ASSP, Password Key Option
	Security EPV with my_ASSP Option
	Security EPVS with my_ASSP Option
	Security EPV with OTP Option
	Security EPVS with OTP Option
	Security EPV with Full OTP Option
	Security EPVS with Full OTP Option
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Background Erase, Program
	SPI Flash Verify Only
	SPI Flash Background Verify Only
	SPI Flash Erase All
	SPI Flash Background Erase All
	SPI Flash Read and Save
	SPI Flash Background Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device

Table 62: MachXO3LF (Continued)

Access Mode	Operation
Slave SPI Interface Programming	Slave SPI Erase, Program, Verify
	Slave SPI Fast Configuration
	Slave SPI Erase, Program, Verify, Secure
	Slave SPI Erase, Program, Verify, Secure Plus
	Slave SPI CFG Erase, Program, Verify
	Slave SPI Erase Only
	Slave SPI Verify Only
	Slave SPI Verify ID
	Slave SPI Display ID
	Slave SPI Display USERCODE
	Slave SPI Read and Save
	Slave SPI Read Status Register
	Slave SPI UFM Program, Verify
	Slave SPI UFM Erase
	Slave SPI Background Erase, Program, Verify
	Slave SPI Background Erase, Program, Verify, Secure
	Slave SPI Background Erase, Program, Verify, Secure Plus
	Slave SPI Background CFG Erase, Program, Verify
	Slave SPI Background Program Feature Rows
	Slave SPI Background Erase, Program, Verify, Feature
	Slave SPI Background Erase Only
	Slave SPI Background Verify Only
	Slave SPI Background SEI Fast Program
	Slave SPI Refresh
	Slave SPI Erase, Program, Verify with Password
	Slave SPI Fast Configuration with Password
	Slave SPI Verify Only with Password
	Slave SPI Erase Only with Password

Table 62: MachXO3LF (Continued)

Access Mode	Operation
I2C Interface Programming	I2C Erase, Program, Verify
	I2C Program
	I2C Erase, Program, Verify, Secure
	I2C Erase, Program, Verify, Secure Plus
	I2C Erase Only
	I2C Verify Only
	I2C Verify ID
	I2C Display ID
	I2C Program Feature Rows
	I2C Recovery Erase Only
	I2C Erase, Program, Verify, Feature
	I2C Background Erase, Program, Verify
	I2C Background Erase, Program, Verify, Secure
	I2C Background Erase, Program, Verify, Secure Plus
	I2C Background Program
	I2C Background Erase Only
	I2C Background Verify Only
	I2C Background Program Feature Rows
	I2C Refresh
	I2C Program Password Key
	I2C Program Password Key with Lock
	I2C Erase Feature Row with Password Key
	I2C EPV with Password Key Option
	I2C EPV with my_ASSP, Password Key Option
	I2C Erase, Program, Verify with Password
	I2C Fast Configuration with Password
	I2C Verify Only with Password
	I2C Erase Only with Password

Return to Top

Table 63: ORCA FPSC

Access Mode	Operation
JTAG 1532 Mode	Program, Verify
	Program
	Verify ID
	Display ID
	Bypass
	Verify USERCODE
	Display USERCODE
	Read and Save
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 64: ORCA4ExxE

Access Mode	Operation
JTAG 1532 Mode	Program, Verify
	Program
	Verify Only
	Verify ID
	Display ID
	Bypass
	Verify USERCODE
	Display USERCODE
	Read and Save
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 65: ORCA3LxxB

Access	Operation
JTAG 1532 Mode	Program, Verify
	Program
	Verify ID
	Display ID
	Bypass
	Verify USERCODE
	Display USERCODE
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 66: ORCA2C/T

Access Mode	Operation
JTAG 1532 Mode	Program
	Bypass
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 67: ORCA3C/T

Access	Operation
JTAG 1532 Mode	Program, Verify
	Program
	Verify ID
	Display ID
	Bypass
	Verify USERCODE
	Display USERCODE
Serial Mode	Serial Program
	Serial Read DONE bit

Return to Top

Table 68: Platform Manager 2

Access Mode	Operation
PTM Programming	PTM Erase, Program, Verify
	PTM Erase, Program, Verify, Secure
	PTM Program
	PTM Erase Only
	PTM Verify Only
	PTM Verify ID
	PTM Display ID
	PTM Display USERCODE
	PTM Read and Save
	PTM Read Status Register
	PTM Read ASC Status
	PTM Read FaultLog Registers
	PTM Read FaultLog Memory
	PTM FaultLog Erase Only
	PTM Erase, Program, Verify, I2CSA
	PTM I2CSA Program
	PTM I2CSA Erase Only
	PTM Bypass
PTM Background Programming	PTM Background Erase, Program, Verify
	PTM Background Erase, Program, Verify, Refresh
	PTM Background Program, Verify, and TransFR
	PTM Background Erase Only
	PTM Background Verify Only
	PTM Background Verify ID
	PTM Background Read and Save
	PTM Background Read Status Register
	PTM Background Read ASC Status
	PTM Background Erase, Program, Verify, I2CSA
	PTM Background Program Feature Rows
	PTM Background Bypass

Table 68: Platform Manager 2 (Continued)

Access Mode	Operation
Flash Programming Mode	FLASH Erase, Program, Verify
	FLASH Erase, Program, Verify, Secure
	FLASH Erase, Program, Verify, Secure Plus
	FLASH Program
	FLASH Verify Only
	FLASH Erase Only
	FLASH Verify ID
	FLASH Display ID
	FLASH Display USERCODE
	FLASH Read and Save
	FLASH Calculate Checksum
	FLASH UFM Program, Verify
	FLASH UFM Erase
	FLASH Verify UFM
	FLASH Read Status Register
	FLASH Secure Device
	FLASH Secure Plus
	FLASH Refresh
	FLASH Bypass

Table 68: Platform Manager 2 (Continued)

Access Mode	Operation
SPI Flash Programming	SPI Flash Erase, Program, Verify
	SPI Flash Background Erase, Program, Verify
	SPI Flash Erase, Program
	SPI Flash Background Erase, Program
	SPI Flash Verify Only
	SPI Flash Background Verify Only
	SPI Flash Erase All
	SPI Flash Background Erase All
	SPI Flash Read and Save
	SPI Flash Background Read and Save
	SPI Flash Calculate File Size Checksum
	SPI Flash Calculate Device Size Checksum
	SPI Flash Bypass
	Scan SPI Flash Device
I2C Interface Programming	I2C Erase, Program, Verify
	I2C Program
	I2C Erase, Program, Verify, Secure
	I2C Erase Only
	I2C Verify Only
	I2C Verify ID
	I2C Display ID
	I2C Program Feature Rows
	I2C Read Status Register
	I2C Erase, Program, Verify, Feature
	I2C Background Erase, Program, Verify
	I2C Background Erase, Program, Verify, Secure
	I2C Background Program, Verify, and TransFR
	I2C Background Program
	I2C Background Erase Only
	I2C Background Verify Only
	I2C Background Program Feature Rows
	I2C Background Read Status Register

Return to Top

See Also

- “Programmer Options” on page 856
- “Device Properties Dialog Box” on page 927

Device Properties Dialog Box

In the Device Properties dialog box you specify the access mode, an operation, and several other features in the General tab. The options available vary depending on the access mode and operation selected. The Device Information tab summarizes the characteristics of the FPGA.

Access Mode Select the mode for programming the device. The following is an alphabetic list of all available access modes with a definition of each operation.

Table 69: Access Mode Descriptions

Access Mode	Description
Advanced Security Encryption File Programming	This mode is used to program the device with an encrypted data file.
Advanced Security File Programming	These operations are used to program the device after the device programmed with the password keys. The password keys is required to shift in first to unlock the device for reconfiguration.
Advanced Security Keys Programming	These operations are used to program the password keys into the device and also program the password keys at the end at the same time.
Advanced Security Production Programming	These operations are used to program the device using the JTAG port and also program the password keys at the end in one shot.
ATDI Pin Selected Programming	This mode is used for devices with an Alternate TDI (ispPAC ProcessorPM and ispPAC Power Manager II).
CRAM Programming	The mode is used for Compressed Random Access Memory (CRAM) configuration.
Dual Boot SPI Flash Programming	This mode is used to program a dual boot hex file into the SPI Flash.
EE	This mode is used for device with Electronically Erasable (EE) memory while the device is not operational.
Feature Row Programming	This mode is used to program, read back, editing the Feature Row sector of the LFMNX or MachXO3D device. The mode also is used to read back, editing the non-volatile Control Register 1 (which also belong to Feature Row sector).
Flash Background Mode	This mode is used to program the embedded flash while the device is still in user mode (background programming) (XFLASH).
Flash Programming Mode	This mode is used to program the embedded Flash while the device is not operational.
I2C Interface Programming	These operations are used to program the device using the I2C port or to generate the files for embedded programming over I2C from a processor.
JTAG 1532 Mode	This mode is used to configure the device through the Joint Test Action Group (JTAG) port.
NVCM Background Programming Mode	This mode is used to program the non-volatile configuration memory (NVCM) while the device is in user mode (Background Mode).

Table 69: Access Mode Descriptions (Continued)

Access Mode	Description
NVCM Programming	This mode is used to program the non-volatile configuration memory (NVCM) while the device is not operational.
I2C NVCM Programming	This mode will program the non-volatile configuration memory (NVCM) through the I2C Port.
SSPI NVCM Programming	This mode will program the non-volatile configuration memory (NVCM) through the Slave SPI Port.
PTM Background Programming	This mode is used to program the background Platform Manager while the device is in user mode (Background Mode).
PTM Programming	This mode is used to program the Platform Manager while the device is not operational.
Serial Mode	This mode is used to program through the Serial Configuration Port.
Slave SPI Interface Programming	These operations are used to program the device using the Slave SPI port.
SSPI Flash Programming	This mode is used to program the device Flash through the Slave SPI port.
SSPI SRAM Programming	This mode is used to program the devices SRAM through the Slave SPI port.
SSPI XSRAM Programming	This mode is used to program the devices SRAM through the Slave SPI port while the device is in user mode (Background Mode).
SPI Flash Background Programming	This mode is used to program the external SPI Flash while the device is in user mode (Background Mode).
SPI Flash Programming	This mode is used to program the external SPI Flash while the device is not operational.
Static RAM Cell Background Mode	The mode is used for SRAM configuration while the device is in user mode (Background Mode).
Static RAM Cell Mode	The mode is used for SRAM configuration while the device is not operational.
X-EE	This mode is used for device with Electronically Erasable (EE) memory while the device is still in user mode (background programming).

Operation After selecting the access mode, select an operation. The following is an alphabetic list of all available operations with a description of each operation.

Table 70: Operation Descriptions

Operation	Description
Blank Check	Checks if the device is erased (blank).
Bypass	No operation is performed on the device.
Calculate Checksum	Calculates the Checksum of the device.
Calculate Data CRC	Calculates the cyclic redundancy check (CRC) of the device.
Calculate Device Size Checksum	Calculates the Checksum of the entire device.

Table 70: Operation Descriptions (Continued)

Operation	Description
Calculate File Size Checksum	Calculates the Checksum of the data contained in the device for the address range of the input file.
Display Control Register0	Reads and Displays the contents of Control Register 0.
Display Control NV Register1	Reads and Displays the content of non-volatile control register 1 which is in the same location of the Feature Row.
Display ID	Reads and Displays the device ID.
Display Programming Pins Status	Reads and Displays the status of the programming pin.
Display Status Register	Reads and Displays the device's Status Register.
Display TraceID	Reads and Displays the user's TraceID.
Display USERCODE	Reads and Displays the USERCODE.
EPS with Flash Protect	EPS (Erase, Program, Secure) after the device programmed with the flash protection password keys. The flash protection password key is required to shift in first to unlock the device for reconfiguration. Secure inhibits read back of the device.
EPS with Flash Protect, Refresh	EPS (Erase, Program, Secure) after the device programmed with the flash protection password keys and then transfers the contents of flash to SRAM. The flash protection password key is required to shift in first to unlock the device for reconfiguration. Secure inhibits read back of the device.
EPV with Flash Protect	EPV (Erase, Program, Verify) after the device programmed with the flash protection password keys. The flash protection password key is required to shift in first to unlock the device for reconfiguration.
EPV with Flash Protect, Refresh	EPV (Erase, Program, Verify) after the device programmed with the flash protection password keys and then transfers the contents of flash to SRAM. The flash protection password key is required to shift in first to unlock the device for reconfiguration.
EPV with Full OTP Option	EPV (Erase, Program, Verify) and enables One time programming (OTP) option. OTP inhibits reprogramming and erasing of the device. This operation mostly used in production programming.
EPV with OTP and my_ASSP Option	EPV (Erase, Program, Verify), enables One time programming option and then program the user's device ID. OTP inhibits reprogramming and erasing of the device. This operation mostly used in production programming.
EPV with Password	EPV (Erase, Program, Verify), after the device programmed with the password keys. The password key is required to shift in first to unlock the device for reconfiguration.
EPV with Password Key Option	EPV (Erase, Program, Verify) the new configuration memory in off-line mode. Then programs the password keys into the device in one shot This operation mostly used in production programming.
EPV with OTP Option	EPV (Erase, Program, Verify) and enables One time programming (OTP) option. OTP inhibits reprogramming and erasing of the device. This operation mostly used in production programming.
EPV with Protect Only Option	EPV (Erase, Program, Verify), and program the flash protection password keys in one shot. This operation mostly used in production programming.

Table 70: Operation Descriptions (Continued)

Operation	Description
EPV with Protect and Lock Option	EPV (Erase, Program, Verify), then program the flash protection password keys and the security lock in one shot. This operation mostly used in production programming.
EPV with my_ASSP Option	EPV (Erase, Program, Verify), and then program the user's device ID in one shot. This operation mostly used in production programming.
EPV with my_ASSP, Password Key Option	EPV (Erase, Program, Verify), and program the user's device ID. Then programs the password keys into the device in one shot. This operation mostly used in production programming.
EPVS with Encrypt Only Option	EPVS (Erase, Program, Verify, Secure), and then program the encryption keys into the device in one shot. Secure inhibits read back of the device. This operation mostly used in production programming.
EPVS with Encrypt and Lock Option	EPVS (Erase, Program, Verify, Secure), then program the encryption keys into the device and locks the device in one shot. Secure inhibits read back of the device. This operation mostly used in production programming.
EPVS with Encryption	EPVS (Erase, Program, Verify, Secure), after the device programmed with the encryption keys. The encrypted data file is required for reconfiguration. Secure inhibits read back of the device.
EPVS with Encryption and Flash Protect	EPVS (Erase, Program, Verify, Secure), after the device programmed with the encryption keys and the flash protect password keys. The encrypted data file is required for reconfiguration. The flash protection password key is required to shift in first to unlock the device for reconfiguration. Secure inhibits read back of the device.
EPVS with Encryption and Flash Protect, Refresh	EPVS (Erase, Program, Verify, Secure), after the device programmed with the encryption keys and the flash protect password keys. And then transfers the configuration from Flash to SRAM. The encrypted data file is required for reconfiguration. The flash protection password key is required to shift in first to unlock the device for reconfiguration. Secure inhibits read back of the device.
EPVS with Flash Protect	EPVS (Erase, Program, Verify, Secure), after the device programmed with the flash protect password keys. The flash protection password key is required to shift in first to unlock the device for reconfiguration. Secure inhibits read back of the device.
EPVS with Flash Protect, Refresh	EPVS (Erase, Program, Verify, Secure), after the device programmed with the flash protect password keys. And then transfers the contents to from Flash to SRAM. The flash protection password key is required to shift in first to unlock the device for reconfiguration. Secure inhibits read back of the device.
EPVS with Full OTP Option	EPVS (Erase, Program, Verify, Secure) and enables One time programming (OTP) option. Secure inhibits read back of the device. OTP inhibits reprogramming and erasing of the device. This operation mostly used in production programming.
EPVS with OTP and my_ASSP Option	EPVS (Erase, Program, Verify, Secure), enables One time programming option and then program the user's device ID. Secure inhibits read back of the device. OTP inhibits reprogramming and erasing of the device.
EPVS with OTP Option	EPVS (Erase, Program, Verify, Secure), enables One time programming (OTP) option. Secure inhibits read back of the device. OTP inhibits reprogramming and erasing of the device. This operation mostly used in production programming.
EPVS with Protect Only Option	EPVS (Erase, Program, Verify, Secure), and then programs the flash protect password keys into the device in one shot. Secure inhibits read back of the device. This operation mostly used in production programming.

Table 70: Operation Descriptions (Continued)

Operation	Description
EPVS with Protect and Lock Option	EPVS (Erase, Program, Verify, Secure), then programs the flash protect password keys and lock for device in one shot. Secure inhibits read back of the device. This operation mostly used in production programming.
EPVS with full feature Option	EPVS (Erase, Program, Verify, Secure), plus program the flash protect password keys and the encryption keys into the device but no feature lock enable fuses. This operation mostly used in production programming.
EPVS with full feature and Lock Option	EPVS (Erase, Program, Verify, Secure), plus program the flash protect keys, the encryption keys and the feature lock enable fuses in one shot. This operation mostly used in production programming.
EPVS with my_ASSP Option	EPVS (Erase, Program, Verify, Secure), and then program the user's device ID. Secure inhibits read back of the device. This operation mostly used in production programming.
Erase	Erases the configuration memory.
Erase All	Erase all configuration in all forms of memory on the device
Erase CFG	Erases the configuration in the Configuration Logic Block (CFG). Leaves all other memory alone.
Erase CFG and EBR Only	Erases the Configuration Logic Block (CFG) and Embedded Block RAM (EBR) only. Leaves all other memory alone.
Erase CFG and UFM Only	Erases the Configuration Logic Block (CFG) and User Fuse Memory (UFM) only. Leaves all other memory alone.
Erase DONE bit	Erase the value of the Done bit. Sets Done bit to 0.
Erase Feature Row	Erases the feature row.
Erase Only	Erases the configuration memory.
Erase Only with Password	Erases the configuration memory after the device programmed with the password keys. The password key is required to shift in first to unlock the device for erasing.
Erase with Flash Protect	Erases the configuration memory after the device programmed with the flash protect password keys. The flash protect password key is required to shift in first to unlock the device for erasing.
Erase Feature Row with Password Key	Erase the contents of the feature row after the device programmed with the password keys. The password key is required to shift in first to unlock the device for erasing.
Erase, Program	Erases the configuration memory and then programs a new configuration into memory.
Erase, Program, Secure	Erase the configuration memory, program a new configuration into memory, and then secures the configuration block. Secure inhibits read back of the device.
Erase, Program, Secure, Refresh	Erase the configuration memory, program a new configuration into memory, secures the configuration block, and then transfers the new configuration over from Flash to SRAM. Secure inhibits read back of the device.
Erase, Program, Verify	Erases, programs, and verifies the new configuration memory. For devices with a Feature Row, this operation will erase, program, and verify the Feature Row in direct programming mode. It will not erase and verify the Feature Row in background programming mode (XFLASH).

Table 70: Operation Descriptions (Continued)

Operation	Description
Erase, Program, Verify, Feature	Erases, programs, and verifies the new configuration memory and the Feature Row.
Erase, Program, Verify, Feature and TransFR	Erases, programs, and verifies the new configuration memory and the Feature Row. Then freezes IO pins and transfers configuration from Flash to SRAM.
Erase, Program, Verify Quad IO SPI	Erases, programs, verifies the new configurations, and then splits up bitstream into four different sectors for Quad IO.
Erase, Program, Verify with OTP	Erases the configuration memory, programs a new configuration, verifies that the configuration is correct, and then sets the one time programming (OTP) option. OTP inhibits reprogramming and erasing of the device.
Erase, Program, Verify with my_ASSP	EPV (Erase, Program, Verify), and then program the user's device ID.
Erase, Program, Verify, I2CSA	Erases the configuration memory, programs a new configuration, verifies that the configuration is correct, and then sets the I2C Slave Address (I2CSA) register in the external ASC.
Erase, Program, Verify, Refresh	Erases the configuration memory, programs a new configuration, verifies that the configuration is correct, and then transfers the contents from Flash to SRAM.
Erase, Program, Verify, Secure	Erases the configuration memory, programs a new configuration, verifies that the configuration is correct, and secures the configuration block. Secure inhibits read back of the device.
Erase, Program, Verify, Secure Plus	Erases the configuration memory, programs a new configuration in to memory, verifies that the contents in memory are correct, and secures both the configuration block and user fuse memory. Secure inhibits read back of the device.
Erase, Program, Verify, Secure, Refresh	Erases the configuration memory, programs a new configuration, verifies that the configuration is correct, secures the configuration block, and then transfers the contents from Flash to SRAM. Secure inhibits read back of the device.
Erase, Program, Verify with Password	Erases the configuration memory, programs a new configuration, and verifies that the configuration is correct after the device programmed with the password keys. The password key is required to shift in first to unlock the device for reconfiguration.
Fast Configuration	Shifts in a bitstream directly into device and the device take care of the rest for configuration.
Fast Configuration with Password	Shifts in a bitstream directly into device and the device takes care of the rest for configuration after the device programmed with the password keys. The password key is required to shift in first to unlock the device for reconfiguration.
Fast Program	Shifts in a bitstream directly into device and the device take care of the rest for configuration.. Also known as burst programming.
Fast Program with Encryption Option	Programs the encryption key into the device and then shifts in the encrypted bitstream directly into device and the device takes care of the rest for configuration. This operation mostly used in production programming.
Fast Program with Password	Shifts in a bitstream directly into device and the device takes care of the rest for configuration after the device programmed with the password keys. The password key is required to shift in first to unlock the device for reconfiguration.
Fast Program, Read and Save	Shifts in a bitstream directly into device and the device take care of the rest for configuration. Then the contents in memory are read and saved to an output file.

Table 70: Operation Descriptions (Continued)

Operation	Description
FaultLog Erase Only	Erase the FaultLog.
FLASH Bypass	Same as “Bypass.”
FLASH CFG Erase Only	Same as “Erase CFG.”
FLASH CFG Erase, Program, Verify	Erases, programs, and verifies the CFG arrays.
FLASH CFG Verify Only	Verify the content of the CFG arrays. Leave all other memory arrays alone.
FLASH Display ID	Same as “Display ID.”
FLASH Display USERCODE	Same as “Display USERCODE.”
FLASH Dry Run	Trigger the downloading the bitstream from CFG array to check internally without actually writing the configuration SRAM.
Flash Erase Only	Erases the contents of Flash memory.
FLASH Erase, Program, Verify	Same as “Erase, Program, Verify.”
FLASH Erase, Program, Verify, Secure	Erases, programs, and verifies the new configuration memory. Locks the local read back and programming of the sectors.
FLASH Read and Save	Same as “Read and Save”
FLASH Read Status Register	Same as “Read Status Register”
FLASH Refresh	Same as “Refresh”
FLASH UFM Erase Only	Erases the User Flash Memory (UFM) arrays. Leaves all other memory arrays alone.
FLASH UFM Erase, Program, Verify	Erases, programs, and verifies the User Flash Memory (UFM) arrays.
FLASH UFM Verify Only	Same as “Verify UFM.”
FLASH Verify ID	Same as “Verify ID.”
FLASH Verify Only	Same as “Verify Only.”
Flash Erase with Flash Protect	Erases the contents of Flash memory after the device programmed with the flash protect password keys. The flash protect password key is required to shift in first to unlock the device for erasing.
I2CSA Erase Only	Erases the value of the I2C Slave Address (I2CSA) register.
I2CSA Program	Sets the value of the I2C Slave Address (I2CSA) register.
Program	Programs a new configuration into memory. Issues programming instructions and then the data are taken from the data file.
Program Control Register0	Programs Control Register 0.
Program Control NV Register1	Program the content of non-volatile control register 1 which is in the same location of the Feature Row.
Program DONE bit	Programs the Done bit. Sets the Done bit to 1.
Program Encryption Key	Programs the encryption key into a device.

Table 70: Operation Descriptions (Continued)

Operation	Description
Program Encryption Key Only	Only programs the encryption key into a device.
Program Encryption Key and Flash Protect	Programs the encryption key and the flash protect password keys.
Program Encryption Key and Flash Protect with Lock	Programs the encryption key, and then sets the encryption lock and the flash protect password keys.
Program Encryption Key with Lock	Programs the encryption key into the device and then sets the lock.
Program Feature Encryption Lock	Sets the Feature Row encryption lock.
Program Feature Row	Programs the Feature Row. Programs the 128-bit feature row data into a device. The setting is from the .fea file. Lock the local “read back” and/or “program” of the FEATURE sector. It will not lock the “erase” of the FEATURE sector.
Program Flash Protect Key Only	Set the Flash Protect Password key.
Program Flash Protect Key with Lock	Sets the Flash Protect Password key, and then sets the key lock so the key cannot be changed again.
Program Key Lock	Set the Key lock so the key cannot be changed again.
Program Only	Programs a new configuration into memory. Issues programming instructions and then the data are taken from the data file.
Program USF from PROM file	Program the TAG/USF memory using a PROM file.
Program Password Key	Programs the password keys into the device.
Program Password Key with Lock	Programs the password keys into the device and locks the devices.
Program and TransFR	Programs the Flash memory, freezes the devices I/O pins and then transfers the contents of non-volatile memory to working memory.
Program, Read and Save	Configures the Flash memory, then reads the contents of that memory and then saves it to a file.
Program, Refresh	Programs the Flash memory of the device, device goes to sleep, and then transfers the configuration from Flash to SRAM.
Program, Secure and TransFR	Programs the Flash memory, secures the configuration block, and freezes the devices I/O pins and then transfers the contents of non-volatile memory to working memory. Secure inhibits read back of the device.
Program TraceID	Programs the TraceID.
Program, Verify	Programs the configuration memory of the device and then verifies the contents of that memory.
Program, Verify, Secure	Programs the configuration memory of a device, verifies the contents of memory, and secures the configuration block. Secure inhibits read back of the device.

Table 70: Operation Descriptions (Continued)

Operation	Description
Program, Verify and TransFR	Programs, Verify the content in memory. Then freezes IO pins and transfers configuration from Flash to SRAM.
RAM Read and Save	Read the contents of SRAM and then writes the contents into a file.
RAM Verify Only	Verifies the contents of SRAM.
Recovery Erase Only	Erase only operation used to recover device. Does not check ID of the device.
Read ASC Status	Reads the status register of an external ASC.
Read DONE bit	Reads to see if the Done bit has or has not been set.
Read Device Properties	Reads the properties of the device.
Read Encryption Key	Reads the contents of encryption key.
Read FaultLog Memory	Read the contents of FaultLog memory.
Read FaultLog Registers	Read the contents of the FaultLog register.
Read Feature Row	Reads the contents of the Feature row.
Read Programming Status	Reads if the Programming Status is pass or fail.
Read TraceID	Reads the user's TraceID from the device.
Read Status Register	Reads the contents of the Status Register.
Read and Save	Reads the contents of memory and saves it to a output file.
Read and Save JEDEC	Reads the contents of memory and saves it to a JEDEC file.
Refresh	Configures the pattern in Non-volatile into SRAM.
Refresh from Flash	Configures the pattern in Flash into SRAM.
Save JEDEC with TAG	Saves to a JEDEC file with TAG memory information.
Scan SPI Flash Device	Scans the SPI Flash in the board.
Secure	Secures the configuration block. Secure inhibits read back of the device.
Secure Device	Secures the configuration block. Secure inhibits read back of the device.
Secure Plus	Secures the configuration block and the user fuse memory. Secure inhibits read back of the device.
Security Read Status Register	Reads the contents of the Status Register for secured device.
Security EPV with Password Key option	EPV (Erase, Program, Verify). For devices with a Feature Row, this operation will erase, program, and verify the Feature Row in direct programming mode. It will not erase and verify the Feature Row in background programming mode (XFLASH). Then programs the password keys into the device. This operation mostly used in production programming.
Security EPV with my_ASSP, Password Key option	EPV (Erase, Program, Verify), and then program the user's device ID. Then programs the password keys into the device. This operation mostly used in production programming.

Table 70: Operation Descriptions (Continued)

Operation	Description
Security EPV with OTP Option	EPV (Erase, Program, Verify), and then performs OTP (One Time Programming) on non-volatile memory and/or OTP (One Time Programming) on SRAM. This operation mostly used in production programming.
Security EPVS with OTP Option	EPV (Erase, Program, Verify), and then Secures the device, performs OTP (One Time Programming) and/or non-volatile memory or OTP on SRAM. This operation mostly used in production programming.
Security EPV with Full OTP Option	EPV (Erase, Program, Verify), and then OTP (One Time Programming) of non-volatile memory, OTP on SRAM and OTP on FEA. This operation mostly used in production programming.
Security EPVS with Full OTP Option	EPV (Erase, Program, Verify), then Secures the device, performs OTP (One Time Programming) of non-volatile memory, OTP on SRAM and OTP on FEA. This operation mostly used in production programming.
Security Erase Encryption Key	Erase the content of AESKEY sector.
Security Erase Public Key	Erase the content of PUBKEY sector.
Security Program Auth Done FlashA	Trigger the authentication validation process for the bitstream store in FlashA without writing configuration data into SRAM. Once the authentication passes, bit “AUTH_DONE” in FlashA will be automatically programmed.
Security Program Auth Done FlashB	Trigger the authentication validation process for the bitstream store in FlashB without writing configuration data into SRAM. Once the authentication passes, bit “AUTH_DONE” in FlashB will be automatically programmed.
Security Program Encryption Key	Programs the 256-bit encryption key into a device. Lock the local “read back” and/or “program” of the AESKEY sector.
Security Program Password Key	Same as “Program Password Key.”
Security Program Public Key	Programs the 512-bit public key into a device. Enable the Authentication mode. Lock the local “read back” and/or “program” of the PUBLICKEY sector. It will not lock the “erase” of the PUBLICKEY sector.
Security Program TraceID	Same as “Program TraceID.”
Security Read Encryption Key (internal only)	Same as “Read Encryption Key.”
Security Read Password Key (internal only)	Read the contents of Password Key.
Security Read Public Key (internal only)	Reads the contents of Public Key.
Security Read TraceID	Same as “Display TraceID.”
Security Read USEC (internal only)	Reads and Displays the contents of USEC sector.
SEI Fast Program	Soft Error Injection fast configuration Programming.
SEI Fast Program with Password	Soft Error Injection fast configuration Programming after the device programmed with the password keys. The password key is required to shift in first to unlock the device for reconfiguration.

Table 70: Operation Descriptions (Continued)

Operation	Description
SPI Flash Calculate Device Size Checksum	Same as “Calculate Device Size Check Sum.”
SPI Flash Erase All	Same as “Erase All.”
SPI Flash Erase, Program, Verify	Same as “Erase, Program, Verify.”
SPI Flash Erase, Program, Verify Quad 1	Same as “Erase, Program, Verify Quad IO SPI”.
SPI Flash Read and Save	Same as “Read and Save.”
SPI Flash Verify Only	Same as Verify Only/Verify the content of external SPI Flash.
SRAM Bypass	Same as “Bypass.”
SRAM Display Control Register0	Same as “Display Control Register0.”
SRAM Display Control Register1	Reads and Displays the contents of Control Register 1.
SRAM Display ID	Same as “Display ID”
SRAM Display USERCODE	Same as “Display USERCODE.”
SRAM Erase Only	Same as “Erase Onl.y”
SRAM Erase, Program, Verify	Same as “Erase, Program, Verify.”
SRAM Fast Configuration	Same as “Fast Configuration”
SRAM Program Control Register0	Same as “Program Control Register0”
SRAM Program Control Register1	Read back, editing and programs Control Register 1.
SRAM Read and Save	Same as “RAM Read and Save.”
SRAM Read Status Register	Same as “Read Status Register.”
SRAM Refresh	Same as “Refresh.”
SRAM Verify ID	Same as “Verify ID.”
SRAM Verify Only	Same as “RAM Verify Only.”
SRAM Verify USERCODE	Same as “Verify USERCODE.”
SSPI Flash Calculate File Size Checksum	Same as “Calculate File Size Check Sum”
TAG Erase	Erases the contents of TAG memory.
TAG Program, Verify	Programs the TAG memory with a new configuration and then verifies the contents of TAG memory.
TAG Read and Program	Reads the contents TAG memory and then programs a new configuration into TAG memory.

Table 70: Operation Descriptions (Continued)

Operation	Description
TAG Verify	Verifies the contents of TAG memory.
TransFR	Freezes IO pins and transfers configuration from Flash to SRAM.
Update Feature Row	Reads and Displays the Feature Row, allow user to modify the content of Feature Row, then program feature row back.
Upload to Static RAM	Upload data to SRAM.
Verify DONE bit	Verify if the Done bit has been set or not.
Verify EBR	Verifies the contents of the Embedded Block RAM (EBR) cells.
Verify Encryption Key	Verifies the Encryption key.
Verify Feature Rows	Verifies the contents of the devices feature row.
Verify ID	Verifies the ID of the device.
Verify Only	Verifies the new configuration memory.
Verify UFM	Verify the contents of User Fuse Memory (UFM).
Verify USERCODE	Verifies the USERCODE of the device.
Verify with Encryption Option	Verifies the new configuration memory and program the encryption key into the device. This operation mostly used in production programming.
Verify Only with Password	Verifies the new configuration memory after the device programmed with the password keys. The password key is required to shift in first to unlock the device for reconfiguration.
XFLASH Bypass	Same as “Bypass.”
XFLASH CFG Erase Only	Same as “Erase CFG.”
XFLASH CFG Erase, Program, Verify	Erases, programs, and verifies the CFG arrays in background mode. The device is still operating during the process.
XFLASH CFG Verify Only	Verify the content of the CFG arrays in background mode. Leave all other memory arrays alone. The device is still operating during the process.
XFLASH Display ID	Same as “Display ID.”
XFLASH Display USERCODE	Same as “Display USERCODE.”
XFLASH Dry Run	Trigger the downloading the bitstream from CFG array to check internally without actually writing the configuration SRAM in background mode. The device is still operating during the process.
XFLASH Erase Only	Same as “Flash Erase Only.”
XFLASH Erase, Program, Verify	Same as “Erase, Program, Verify.”
XFLASH Erase, Program, Verify, Secure	Erases, programs, and verifies the CFG arrays. Locks the local read back of the Flash sectors in background mode. The device is still operating during the process.
XFLASH Program and TransFR	Same as “Program Verify and TransFR.”
XFLASH Read and Save	Same as “Read and Save.”

Table 70: Operation Descriptions (Continued)

Operation	Description
XFLASH Read Status Register	Same as “Read Status Register.”
XFLASH Refresh	Same as “Refresh.”
XFLASH TransFR	Freezes IO pins and transfers configuration from Flash to SRAM.
XFLASH UFM Erase Only	Erases the User Flash Memory (UFM) arrays in background mode. Leaves all other memory arrays alone. The device is still operating during the process.
XFLASH UFM Erase, Program, Verify	Erases, programs, and verifies the User Flash Memory (UFM) arrays in background mode. The device is still operating during the process.
XFLASH UFM Verify Only	Same as “Verify UFM.”
XFLASH Verify ID	Same as “Verify ID.”
XFLASH Verify Only	Same as “Verify Only.”
XSRAM Bypass	Same as “Bypass.”
XSRAM Display Control Register0	Same as “Display Control Register0.”
XSRAM Display Control Register1	Reads and Displays the content of Control Register 1 in background mode.
XSRAM Display ID	Same as “Display ID.”
XSRAM Display USERCODE	Same as “Display USERCODE.”
XSRAM Read and Save	Same as “RAM Read and Save.”
XSRAM Read Status Register	Same as “Read Status Register.”
XSRAM Refresh	Same as “Refresh.”
XSRAM SEI Fast Program	Same as “SEI Fast Program.”
XSRAM Verify ID	Same as “Verify ID.”

Programming File Selects the data file for programming or verify.

Use I2C Interface of the Lattice HE-USBN-2B Download Cable Check this option to program the device using the I2C port with the Lattice USB-2B download cable.

I2C Slave Address Enter I2C slave address in binary.

Reinitialize Part on Program Error Reinitializes the part to its blank, or erased, state when a programming error occurs.

TCK Frequency Specifies the TCK clock frequency for the chosen device. This value is used to determine the TCK clock period and the length of the delay times.

SVF Vendor Select the SVF vendor for the chosen device.

Instruction Register Length Specify the instruction register length for putting the device into BYPASS during programming. The software automatically fills in this box when you select a BSDL file. You must enter the instruction register length for a JTAG-NOP device if a BSDL file is not selected.

Load from File Import the IR length from a BSDL or SVF file.

Reinitialize Part on Program Error Reinitializes the device to its blank, or erased, state when a programming error occurs.

Verify Readback Bitstream File Select a custom verify mask file.

Family Select an external memory family from the drop-down list.

Vendor Select an external memory vendor from the drop-down list.

Device Select an external memory device from the drop-down list.

Package Select an external memory package from the drop-down list.

Data File Size Displays the full file size. You can change this size by typing a different file size. This feature is useful if you only want to use a portion of the file for the operation. If you change the file size, it must be no larger than the full file size or the device size.

Start Address Displays the start address of the flash device. For the AMD parallel flash, the starting address is automatically selected and cannot be changed.

End Address Displays the end address of the flash.

Erase SPI on Programming Error Erases the flash device if a verify error occurs.

Secure SPI Flash Golden Pattern Sectors Allows user to secure second half of SPI Flash to secure golden pattern. Available only on devices that support this secure feature.

Enter Key Enter encryption key.

Confirm Key Confirm encryption key.

Load Key Load an encryption key from an external file.

Save Key Save encryption key to external file.

Format Choose between ASCII or Hex key display.

Show Key Toggles display of key.

Password Protection Options (Provide key file if password protection enabled.) If device password protection is enabled, User must provide password key file.

Flash-A Programming Options Allows user to select A target flash block.

Flash-B Programming Options Allows user to select B target flash block.

Flash-C Programming Options Allows user to select C target flash block. Flash C includes UFM2 and UFM3 flash blocks which require programming files.

Device Information Tab The Device Information tab displays information on the device selected. The content can vary depending upon the device selected and can include:

- Family Name
- Device Name
- VCC Voltage Supply
- VCCJ Voltage Supply
- Address Register Length
- Data Register Length
- JTAG IDCODE
- JTAG IDCODE MASK
- IDCODE Length
- SOFT IDCODE
- Instruction Register Length
- USERCODE Register Length
- Boundary Scan Register Length
- Support External ASC Device
- Support FPGA Loader
- Maximum Erase Pulse Width (ms)
- Maximum Program Pulse Width (ms)

See Also

- “Programmer Options” on page 856
- “Device Properties Dialog Box—Device, Access Mode, and Operation Options” on page 857

Edit Custom Device Dialog Box

The following options are available in the Edit Custom Device dialog box:

Device Family Dropdown list of custom device families.

Device The name that was assigned to the device.

Package The name that was assigned to the package.

Device Description The description that was assigned to the device.

See Also

- "Programmer Options" on page 856

Custom SPI Flash Dialog Box

The following options are available in the Custom SPI Flash dialog box:

Device Family Dropdown list of custom device families.

Device Description The name that was assigned to the device.

Device Name This can be any alpha-numeric string that describes the device name (for example: SPI-M25P32).

Package This can be any alpha-numeric string that describes the package (for example: 16-pin SOIC).

Device Vendor The vendor of the custom SPI Flash device.

Device ID This can be any alpha-numeric string that describes the device ID (for example: 0x15).

Byte Per Sector Allows you to choose desired bytes per sector value from dropdown.

Protection Options ON If the SPI Flash device supports sector protection, check this box to enable protection options. This will allow you to select **Secure SPI Flash Golden Pattern Sectors** in the "Device Properties Dialog Box" on page 927.

See Also

- "Programmer Options" on page 856

BSCAN Configuration Dialog Box

The following options are available in the BSCAN Configuration dialog box:

BSCAN Configuration Type Allows you to choose one of five BSCANs configuration types: BSCAN2 4 Port, BSCAN2 8 Port Asset Model, BSCAN2

8 Port JTAG/Corelis Model, BSCAN2 12 Port JTAG/Corelis Model, or BSCAN2 16 Port JTAG/Corelis Model.

Select Ports Select the scan ports for the chains you want to access.

Leave Selected Ports In Select RTI, run test idle, to configure the BSCAN2. Selecting TLR, test logic rest, will remove the selected ports from the configuration.

Reset All Ports Resets all ports to the TLR state and the interface to the default state. Select this button to begin another configuration session.

Reset Selected Ports Resets the selected ports to the TLR state.

Configure Selected Ports Scans the selected ports and displays the chains and devices for programming.

See Also

- "Programmer Options" on page 856

Settings Dialog Box

The following options are available in the Settings dialog box. In the General tab, you can specify several options. In the Programming tab, you can specify aspect of how the programming is done.

At Programmer Start-Up Allows you to select between showing the Programmer Getting Started dialog box at start-up, or opening the last Programmer project at start-up.

USERCODE Display Allows you to choose in which format the USERCODE will be displayed. Choices are Hex, ASCII, and Decimal.

Device Family Selection List Order Allows you to display the Device Family drop-down list in the Programmer main window in either chronological or alphabetical order. The default is chronological order.

Log File Path Shows the location of the Programmer log file, and allows you to browse to and set a new location for the Programmer.log file.

Clear Log File Each Time Application Starts Clears log file each time the application starts: If the box is unchecked, the log file will continue increase in size until the file is manually erased.

Turbo Mode Programs all the devices on the board in parallel. If an Operation Override is selected, all Operation descriptions in the chain file are temporarily changed to the override setting.

Sequential Mode Programs all the devices on the board one at a time. If an Operation Override is selected, all Operation descriptions in the chain file are temporarily changed to the override setting.

Use Default JTAG States (TLR/TLR) Uses Test-Logic-Reset (TLR) as the starting and ending JTAG state of the JTAG State Machine for download.

Avoid Test Logic Reset (TLR) State Avoids the Test Logic Reset State of the JTAG State Machine during download.

Use Custom JTAG States Specify Test-Logic-Reset (TLR) or Run-Test/Idle (RTI) as the starting and ending JTAG state of the JTAG State Machine for download.

Initial TAP State Specify TLR or RTI as the starting JTAG state of the JTAG State Machine for download.

Final TAP State Specify TLR or RTI as the ending JTAG state of the JTAG State Machine for download.

Check Cable Setup Before Programming Confirms that the cable signals are correctly connected to the board and devices in the JTAG chain on the board match the devices selected in the .xcf file.

Continue Download on Error Ignores any errors while downloading and continues running.

See Also

- "Programmer Options" on page 856

Cable Settings Tab

The following options are available in the Cable Settings tab:

Detect Cable Automatically detects the parallel port to which the download cable is connected.

Cable Specifies the download cable type:

- **HW-USBN-2A—**(Lattice HW-USBN-2A USB port programming cable)
- **HW-USBN-2B (FTDI)**—(Lattice HW-USBN-2B (FTDI) USB programming cable)
- **HW-DLN-3C (Parallel)**—(Lattice HW-DLN-3C parallel programming cable)

Port Specifies the serial port to which the download cable is connected. Select the port from the list or click Detect Cable.

Custom Port (Hex) Specifies a custom parallel port to which the download cable is connected. Use hexadecimal format to type the port name.

Use Default Clock Divider Uses the fastest TCK clock speed.

Use Custom Clock Divider Enables the Use Custom Clock Divider feature.

TCK Divider Setting (0 - 10x) Allows you to slow down the TCK clock. This is done by extending the low period of the clock. Refer the following tables for specific frequency settings for USB-2B (2232H FTDI USB host chip), USB-2B (2232D FTDI USB host chip), and USB-2A and Parallel port cables.

Table 71: USB-2B (2232H FTDI USB host chip)

Divider	Clock Frequency[1]	Divider	Clock Frequency[1]
0	30 MHz	5	5 MHz
1	15 MHz (Default)	6	4.2 MHz
2	10 MHz	7	3.7 MHz
3	7.5 MHz	8	3.1 MHz
4	6 MHz	9	3 MHz
		10	2.7 MHz

[1]Calculation formula for USB-2B (2232H FTDI USB host chip):

Frequency = 60 MHz / (1 + ClockDivider) *2

Table 72: USB-2B (2232D FTDI USB host chip)

Divider	Clock Frequency[2]	Divider	Clock Frequency[2]
0	6 MHz	5	1 MHz
1	3 MHz (Default)	6	0.8 MHz
2	2 MHz	7	0.7 MHz
3	1.5 MHz	8	0.65 MHz
4	1.2 MHz	9	0.6 MHz
		10	0.5 MHz

[2]Calculation formula for USB-2B (2232D FTDI USB host chip):

Frequency = 12 MHz / (1 + ClockDivider) *2

Table 73: USB-2A and Parallel port

Divider	Low Pulse Width delay[3]	Divider	Low Pulse Width delay[3]
0	NA	5	5x
1	1x	6	6x
2	2x	7	7x
3	3x	8	8x
4	4x	9	9x
		10	10x

[3]The USB-2A frequency fixed at 1.5 MHz and Parallel Port frequency fixed at 500 kHz

Use Default I/O Settings Only use the four JTAG signals.

Use Custom I/O Settings Specify additional non-JTAG signals connected to the board.

INITN Pin Connected Select this option if you connect the INIT pin to the download cable. This option is only available with the Lattice HW-USBN-2A USB port programming cable.

Done Pin Connected Select this option if you connect the DONE pin to the download cable. This option is not available for the Lattice HW-DLN-3C parallel programming cable.

TRST Pin Connected Select this option if you connect the TRST pin to the download cable. Specify active high or active low.

PROGRAMN Pin Connected Select this option if you connect the PROGRAMN pin to the download cable.

ispEN Pin Connected Select this option if you connect the ispEN pin to the download cable. Specify active high or active low.

See Also ▶"Programmer Options" on page 856

Firmware Tab

The following options are available in the Firmware tab.

MachXO2 firmware Select a file to update the firmware on the FTDI cable's MachXO2.

POWR607 firmware Select a file to update the firmware on the FTDI cable's POWR607.

Update Firmware Reprograms the firmware of the device/devices on cable.

Save Existing Firmware Users can save current cable's firmware on a file.

Cable's Firmware Version Shows the cable's firmware version number.

See Also ▶"Programmer Options" on page 856

Edit I/O State Dialog Box

The following options are available in the Edit I/O State dialog box:

I/O State The state of the I/O pin (input or output, high or low).

BSDL File A BSDL file, which is the acronym for Boundary Scan Description Language file, is a subset of Very High Speed IC Hardware Description Language (VHDL) that describes the boundary scan device package, pin description and boundary scan cell of the input and output pins. The BSDL file is very important in generating a boundary scan test. Without a BSDL file, it is impossible to generate any boundary scan test.

Package The target device package.

See Also ▶"Programmer Options" on page 856

Display Status Register Dialog Box

The following options are available in the Display Status Register dialog box:

Refresh Refreshes the status of the device.

See Also "Programmer Options" on page 856

Custom Device Information Dialog Box

The following options are available in the Custom Device Information dialog box:

Device Name Edit box limited to 12 characters maximum. Valid characters are alphanumeric and a dash.

Device Full Name Edit box limited to 12 characters maximum. Valid characters are alphanumeric and a dash.

Package This can be any alpha-numeric string that describes the package (for example: 16-pin SOIC).

ID Length ID length equal to 32, per IEEE 1149.1 Std. This box is not editable.

Device ID (Hex) Limited to four hexadecimal characters. Valid characters are “Ox” followed by two hexadecimal characters. Default value is OxFFFFFFFF.

Instruction Register Length Select any number ranging from 2 to 32.

Read ID Instruction Must be entered in binary only (0 or 1). The number of characters entered is limited to the Instruction Register Length.

Instruction Bypass Pattern Must be entered in binary only (0 or 1). The number of characters entered is limited to the Instruction Register Length.

BSDL File Browse to the desired bsm file. A BSDL file, which is the acronym for Boundary Scan Description Language file, is a subset of Very High Speed IC Hardware Description Language (VHDL) that describes the boundary scan device package, pin description and boundary scan cell of the input and output pins. The BSDL file is very important in generating a boundary scan test. Without a BSDL file, it is impossible to generate any boundary scan test.

See Also ▶“Programmer Options” on page 856

Cable Signal Tests Dialog Box

The following options are available in the Cable Signal Tests dialog box:

Toggle Alternates between log 1, logic 0, and logic 1.

Hold High Holds at logic 1.

Hold Low Holds at logic 0.

Test Applies the selected settings to the selected pin connections. This causes the pins that have a toggle setting to toggle once from logic 1 to logic 0 to logic 1.

Loop Test Performs continuous testing, causing the pins that have a toggle setting to toggle continuously until you click ESC.

View Log Opens the log file, an ASCII text file that summarizes the results of the Cable Signal Test operations.

Power Check Tests the cable and power connections, causing the VCC Status LED to blink green or red to indicate that cable and power are detected or not detected.

VCC Status Shows the status of the cable and power connection. The LED blinks green when the cable and power are detected. It blinks red and produces an error message when cable or power is not detected.

Toggle Delay Sets the delay after each operation. Default is 0.

Number of Bytes Sets the number of bytes read by each operation. Default is 0.

See Also

- "Programmer Options" on page 856

Deploying the Design with the Deployment Tool

The Deployment Tool allows you to generate files for deployment for single devices, a chain of devices, and can also convert data files to other formats and use the data files it produces to generate other data file formats.

Deployment Tool is a stand-alone tool available from the Diamond Accessories. The Deployment Tool graphical user interface (GUI) is separate from the Diamond design environment.

A four-step wizard allows you to select deployment type, input file type, and output file type.

See Also

- "Deployment Function Types" on page 950
- "Input File Descriptions" on page 993
- "Input File Formats" on page 994
- "Bitstream Output File Descriptions" on page 995
- "Bitstream Output Formats" on page 996
- "SVF, ISC, JEDEC, and STAPL Output Formats" on page 996
- "Output Options Descriptions" on page 997
- "Creating a New Deployment" on page 997
- "Opening an Existing Deployment" on page 999
- "Encrypting a JEDEC with Deployment Tool" on page 1000
- "Opening an Existing Deployment While Running the Deployment Tool" on page 1004
- "Creating a New Deployment While Running the Deployment Tool" on page 1005
- "Using Quick Launch Button to Create a New Deployment" on page 1005
- "Viewing the Deployment Tool Log File" on page 1007
- "Changing the Deployment Tool Log File Settings" on page 1007

Deployment Function Types

This section provides tables that list device family, input file types, output file extensions, and options for each available deployment function type.

There are four types of deployments available. Click on the list items below to jump to topics that contain lists of device family, input file types, output file type/extensions, and options for each deployment function type.

- "File Conversion Deployment Function Type" on page 951
- "Tester Deployment Function Type" on page 958

- "Embedded System Deployment Function Type" on page 964
- "External Memory Deployment Type" on page 978

File Conversion Deployment Function Type

This topic provides tables that list device family, input file types, output file extensions, and options for File Conversion deployment function type.

IEEE 1532 ISC Data File—Device Family, Input File 1, and Output File

The following table lists device family, input file types, and output file extensions for IEEE 1532 ISC data file conversion deployment.

Table 74:

Device Family	Input File 1	Output File
spMACH4000/ ZE ispXPGA Mature	JEDEC (.jed)	ISC Data (.isc)

IEEE 1532 ISC Data File—Options

The following table lists Options for IEEE 1532 ISC data file conversion deployment.

Table 75:

Device Family	Overwrite USERCODE		
ispMACH4000/ ZE ispXPGA Mature	**Off (Default)**	**On**	**Checksum**
	Input File USERCODE.	User specified value.	JEDEC checksum.

Application Specific BSDL File—Device Family, Input File 1, Input File 2, and Output File

The following table lists device family, input file types, and output file extensions for Application Specific BSDL data file conversion deployment.

Table 76:

Device Family	Input File 1	Input File 2	Output File
Lattice	JEDEC (.jed) Argument File (.alt)	Generic BSDL (.bsm)	BSDL File (.bsm)

Application Specific BSDL File—Options

The following table lists options for Application Specific BSDL data file conversion deployment.

Table 77:

Device Family	Convert Bi-directional I/O's to Input and Output	
Lattice	**Off (Default)**	**On**
	Do not convert.	Convert.

JEDEC File—Device Family, Input File 1, and Output File

The following table lists device family, input file types, and output file extensions for JEDEC file conversion deployment.

Table 78:

Device Family	Input File 1	Output File
LFMNX LIFMD LIFMDF MachXO2 MachXO3D MachXO3L Platform Manager Platform Manager 2	JEDEC (.jed)	JEDEC (.jed)
LatticeXP2 LatticeXP MachXO	JEDEC (.jed) Bitstream (.bit, .rbt)	
ispXPGA Mature	JEDEC (.jed) ISC Data (.isc)	

JEDEC File—Options—Overwrite USERCODE

The following table lists options—Overwrite USERCODE—for JEDEC file conversion deployment.

Table 79:

Device Family	Overwrite USERCODE			
	Off (Default)	**On**	**Checksum**	**Disable (Internal Command Line Only)**
Platform Manager 2 Platform Manager LatticeXP2 LatticeXP LFMNX LIFMD LIFMDF MachXO2 MachXO3D MachXO3L MachXO ispXPGA Mature	Input File USERCODE.	User specified value.	JEDEC checksum.	Omit USERCODE from JEDEC.

JEDEC File—Options—Program Security Bit

The following table lists options—Program Security Bit—for JEDEC file conversion deployment.

Table 80:

Device Family	Program Security Bit		
	Default (Default)	**Off**	**On**
Platform Manager 2 Platform Manager LatticeXP LFMNX LIFMD LIFMDF MachXO2 MachXO3D MachXO3L ispXPGA Mature	Use setting from input file.	Do not program security fuses.	Program security fuses.

JEDEC File—Options—Encryption

The following table lists options—Encryption—for JEDEC file conversion deployment.

Table 81:

Device Family	Encryption
LFMNX MachXO3D LatticeXP2	**Encryption Key** Load from BEK file.

Bitstream—Device Family, Input File 1, Format, and Output File

The following table lists device family, input file types, format, and output file extensions for bitstream conversion deployment.

Table 82:

Device Family	Input File 1	Format	Output File
LFMNX MachXO3D ECP5 LatticeECP3 LatticeECP2/MS LatticeECP2/M LatticeECP LatticeSC/M ORCA Unknown or Non-Lattice	Bitstream (.bit, .rbt).	Binary Bitstream	Bitstream (.bit)
		ASCII Bitstream	Bitstream (.rbt)
		Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tektronix Hex	Hex (.xtek)
LFMNX MachXO3D LatticeXP2 LatticeXP MachXO2	JEDEC (.jed) Bitstream (.bit, .rbt)	Binary Bitstream	Bitstream (.bit)
		ASCII Bitstream	Bitstream (.rbt)
		Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tektronix Hex	Hex (.xtek)

Bitstream—Options—Encryption

The following table lists options—Encryption—for bitstream conversion deployment..

Table 83:

Device Family	Encryption	
	Configuration Mode	**Encryption Key**
		Load from BEK File
LatticeECP3 LatticeECP2S/MS	Selects the appropriate configuration mode for the bitstream encryption.	Load from BEK file.

Bitstream—Options—Encryption

The following table lists options—Encryption—for bitstream conversion deployment..

Table 84:

Device Family	Encryption
ECP5	Encryption Key
LFMNX MachXO3D	Load from BEK File

Bitstream—Options—Authentication

The following table lists options—Authentication—for bitstream conversion deployment..

Table 85:

Device Family	Authentication
LFMNX MachXO3D	Public Key and Private Key or Signature Key
	Load from GUI and load Public Key (.pub) File and Private File (.prv).

Bitstream Options—Verify ID Code, Frequency, Compression

The following table lists options—Verify ID Code, Frequency, and Compression—for bitstream conversion deployment.

Figure 9:

Device Family	Verify ID Code			Frequency		Compression		
	Default (Default)	On	Off	Default (Default)	Frequency Selection	Default (Default)	On	Off
ECP5 LFMNX MachXO2 MachXO3D MachXO3L	Use setting from input file.	Include verify ID in bitstream .	Do not verify ID code.	Use setting from input file.	Valid frequency device dependent.	Use setting from input file.	Compress bitsream data.	Do not compress.

Figure 9:

Device Family	Verify ID Code			Frequency		Compression		
LatticeECP3 LatticeXP LatticeXP2	Use setting from input file.	Include verify ID in bitstream .	Do not verify ID code.	Use setting from input file.	Valid frequency device dependent.	-	-	-
LatticeSC/M	-	-	-	-	-	Use setting from input file.	Compress bitsream data.	Do not compress.

Bitstream Options—CRC Calculation, Overwrite USERCODE, Byte Wide Bit Mirror, and Retain Bitstream Header

The following table lists options—CRC Calculation, Overwrite USERCODE, Byte Wide Bit Mirror, and Retain Bitstream Header—for bitstream conversion deployment.

Figure 10:

Device Family	CRC Calculator			Overwrite USERCODE		Byte Wide Bit Mirror (Intel Hex (.mcs), Motorola Hex (.exo), and Extended Tektronix Hex (.xtek) only)		Retain Bitstream Header (Intel Hex (.mcs), Motorola Hex (.exo), and Extended Tektronix Hex (.xtek) only)	
	Default (Default)	**On**	**Global CRC Only**	**Off (Default**	**On**	**Off (Default)**	**On**	**Off (Default)**	**On**
ECP5 LatticeECP3 LatticeXP2 LFMNX MachXO2\ MachXO3D MachXO3L	Use setting from input file.	Include CRC for each data frame.	Include global CRC only.	Input File USERCODE.	User specified value.	No change to bit order.	Flips each byte.	Replace bitstream header.	Retain bitstream header.
LatticeECP2S/MS LatticeECP2/M	Use setting from input file.	Include CRC for each data frame.	Include global CRC only.	Input File USERCODE.	User specified value.	No change to bit order.	Flips each byte.	-	-
LatticeEC/P LatticeSC/M LatticeXP	-	-	-	Input File USERCODE.	User specified value.	No change to bit order.	Flips each byte.	Replace bitstream header.	Retain bitstream header.

Figure 10:

Device Family	CRC Calculator			Overwrite USERCODE		Byte Wide Bit Mirror (Intel Hex (.mcs), Motorola Hex (.exo), and Extended Tektronix Hex (.xtek) only)		Retain Bitstream Header (Intel Hex (.mcs), Motorola Hex (.exo), and Extended Tektronix Hex (.xtek) only)	
ORCA	-	-	-	-	-	No change to bit order.	Flips each byte.	Replace bitstream header.	Retain bitstream header.
Unknown or Non-Lattice	-	-	-	-	-	No change to bit order.	Flips each byte.	-	-

JEDEC to Hex File Conversion

The following table lists device family, input file types, format, and output file extensions for JEDEC to Hex deployment.

Table 86:

Device Family	Input File 1	Format	Output File
Lattice	JEDEC (.jed)	ASCII Raw Hex	ASCII Hex (.hex)
		Binary Raw Hex	Binary Hex (.bin)

Tester Deployment Function Type

This topic provides tables that list device family, input file types, output file extensions, and options for Tester deployment function type.

SVF Single Device—Device Family, Input File 1, and Output File

The following table lists device family, input file types, and output file extensions for SVF single device deployment.

Table 87:

Device Family	Input File 1	Output File
Lattice Single Device	JEDEC (.jed) ISC Data (.isc) Bitstream (.bit, .rbt)	SVF File (.svf)

SVF Single Device—Options—Operation

The following table lists options—Operation—for SVF single device deployment.

Table 88:

Device Family	Operation
Lattice Single Device	Device dependent list.

SVF Single Device—Options—Write Header and Comments, Write Rev D Standard SVF File

The following table lists options—Write Header and Comments, Write Rev D Standard SVF File—for SVF single device deployment.

Table 89:

Device Family	Write Header and Comments		Write Rev D Standard SVF File	
Lattice Single Device	**On (Default)**	**Off**	**Off (Default)**	**On**
	Write headers and comments.	Omit header and comments.	Rev E Standard SVF For Flash based FPGA's, Lattice Extended SVF.	Rev D Standard SVF.

SVF Single Device—Options—Use RUNTEST from Rev C; For Erase, Program, and Verify Operations, Skip Verify

The following table lists options—Use RUNTEST from Rev C; For Erase, Program, and Verify Operations, Skip Verify—for SVF single device deployment.

Table 90:

Device Family	Use RUNTEST from Rev C		For Erase, Program, and Verify Operations, Skip Verify	
Lattice Single Device	**Off (Default)**	**On**	**Off (Default)**	**On**
	Standard RUNTEST.	Rev C RUNTEST.	Include verify in EPV operations.	Omit verify in EPV operations.

SVF Single Device—Options—Include RESET at the End of the SVF File; Set Maximum Data Size per Row (Kbits)

The following table lists options—Include RESET at the End of the SVF File; Set Maximum Data Size per Row (Kbits)—for SVF single device deployment

Table 91:

Device Family	Include RESET at the End of the SVF File		Set Maximum Data Size per Row (Kbits)	
Lattice Single Device	**Off (Default)**	**On**	**Off (Default)**	**On**
	Do not write RESET at the end of the SVF file.	Write RESET at the end of the SVF file.	-	8/16/32/64/128/256

SVF JTAG Chain—Device Family, Input File 1, and Output File

The following table lists device family, input file types, and output file extensions for SVF JTAG chain deployment.

Table 92:

Device Family	Input File 1	Output File
Lattice Chain	XCF File (.xcf)	SVF File (.svf)

SVF JTAG Chain—Options—Operation

The following table lists options—Operation—for SVF JTAG chain deployment.

Table 93:

Device Family	Operation
Lattice Single Device.	Contained in XCF file.

SVF JTAG Chain—Options—Write Header and Comments, Write Rev D Standard SVF File

The following table lists options—Write Header and Comments, Write Rev D Standard SVF File—for SVF JTAG chain deployment.

Table 94:

Device Family	Write Header and Comments		Write Rev D Standard SVF File	
Lattice Single Device	**On (Default)**	**Off**	**Off (Default)**	**On**
	Write headers and comments.	Omit header and comments.	Rev E Standard SVF. For Flash based FPGA's, Lattice Extended SVF.	Rev D Standard SVF.

SVF JTAG Chain—Options—Use RUNTEST from Rev C; For Erase, Program, and Verify Operations, Skip Verify

The following table lists options—Use RUNTEST from Rev C; For Erase, Program, and Verify Operations, Skip Verify—for SVF JTAG chain deployment.

Table 95:

Device Family	Use RUNTEST from Rev C		For Erase, Program, and Verify Operations, Skip Verify	
Lattice Single Device	**Off (Default)**	**On**	**Off (Default)**	**On**
	Standard RUNTEST.	Rev C RUNTEST.	Include verify in EPV operations.	Omit verify in EPV operations.

SVF JTAG Chain—Options—Include RESET at the End of the SVF File; Set Maximum Data Size per Row (Kbits)

The following table lists options—Include RESET at the End of the SVF File; Set Maximum Data Size per Row (Kbits)—for SVF JTAG Chain deployment

Table 96:

Device Family	Include RESET at the End of the SVF File		Set Maximum Data Size per Row (Kbits)	
Lattice Single Device	**Off (Default)**	**On**	**Off (Default)**	**On**
	Do not write RESET at the end of the SVF file.	Write RESET at the end of the SVF file.	-	64/8/16/32/128/256

STAPL Single Device—Device Family, Input File 1, and Output File

The following table lists device family, input file types, and output file extensions for STAPL single device deployment.

Table 97:

Device Family	Input File 1	Output File
Lattice Single Device.	JEDEC (.jed)	STAPL File (.stp)
	ISC Data (.isc)	

STAPL Single Device—Options—ACA Compression; Include Print Statements; For Erase, Program, and Verify Operations, Skip Verify

The following table lists options—ACA Compression; Include Print Statements; For Erase, Program, and Verify Operations, Skip Verify—for STAPL single device deployment.

Table 98:

Device Family	ACA Compression		Include Print Statements		For Erase, Program, and Verify Operations, Skip Verify	
Lattice Single Device	**On (Default)**	**Off**	**Off (Default)**	**On**	**Off (Default)**	**On**
	Compress data.	Uncompressed data.	Print statements as comments.	Include print statements.	Include verify in EPV operations.	Omit verify in EPV operations.

STAPL JTAG Chain—Device Family, Input File 1, and Output File

The following table lists device family, input file types, and output file extensions for STAPL JTAG chain deployment.

Table 99:

Device Family	Input File	Output File
Lattice Chain	XCF File (.xcf)	STAPL File (.stp)

STAPL JTAG Chain—Options—ACA Compression; Include Print Statements; For Erase, Program, and Verify Operations, Skip Verify

The following table lists options—ACA Compression; Include Print Statements; For Erase, Program, and Verify Operations, Skip Verify—for STAPL single device deployment.

Table 100:

Device Family	ACA Compression		Include Print Statements		For Erase, Program, and Verify Operations, Skip Verify	
Lattice Single Device	**On (Default)**	**Off**	**Off (Default)**	**On**	**Off (Default)**	**On**
	Compress data.	Uncompressed data.	Print statements as comments.	Include print statements.	Include verify in EPV operations.	Omit verify in EPV operations.

ATE—Device Family, Input File 1, and Output File

The following table lists device family, input file types, and output file extensions for ATE deployment.

Note

The HP3070 and HP3065 output files are Agilent Pattern Capture Format File (.pcf) format.

Table 101:

Device Family	Input File 1	Output File
Lattice Chain	XCF File (.xcf)	Generic Vector Format (.tst)
		GenRad (.gr)
		HP3070 (.pcf)
		HP3065 (.pcf)
		Teradyne 1800 (.asc)
		Teradyne L200/300 (.asc)

ATE—Options—Tester Type

The following table lists options—Tester Type—for ATE deployment.

Table 102:

Device Family	Operation
Lattice Chain	Generic Vector Format
	GenRad
	HP3070
	HP3065
	Teradyne 1800
	Teradyne L200/300

ATE—Options—ACA Compression; Include Print Statements; For Erase, Program, and Verify Operations, Skip Verify

The following table lists options—Vector in ispDCD Format; For Erase, Program, and Verify Operations, Skip Verify; Header file—for ATE deployment.

Table 103:

Device Family	Vector in ispDCD Format		For Erase, Program, and Verify Operations, Skip Verify		Header File (HP3070, HP3065, Teradyne 1800, and Teradyne L200/ 300 only)	
Lattice Chain	**Off (Default)**	**On**	**Off (Default)**	**On**	**Blank (Default)**	**User Selected Header File**
	Standard format.	ispDCD format.	Include verify in EVP operations.	Omit verify in EVP operations.	Use default header file.	User customer header file.

ATE—Options—Split File

The following table lists options—Split File—for ATE deployment.

Table 104:

Device Family	Split File							
Lattice Chain	**Off (Default)**	**On**	**Initialize Each Split File**		**Max Number of Vectors per File**		**Split File At (HP3070 and HP3065 only)**	
	No Split	Split	**Off (Default)**	**On**	**1000 (Default)**	**Enter Value from GUI**	**Active High (Default)**	**Active Low**
			Split without initialization.	Initialize each time after split.	1000 vectors per file.	Enter value from GUI.	Split at active high.	Split at active low.

See Also ▶"Input File Descriptions" on page 993

Embedded System Deployment Function Type

This topic provides tables that list device family, input file types, output file extensions, and options for Embedded System deployment function type.

JTAG Full VME Embedded—Device Family, Input XCF FIe, and Output File

The following table lists device family, input file types, and output file extensions for JTAG Full VME embedded deployment

Table 105:

Device Family	Input XCF File	Output File
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M Platform Manager Platform Manager 2 LatticeXP2 LatticeXP LFMNX MachXO3D MachXO3L MachXO2 MachXO ispXPGA ORCA ispMACH4000 / ZE ispPAC-CLK5300 ispPAC-CLK5400 ispPAC-POWR605 ispPAC-POWR607 ispPAC-POWR6AT6 ispPAC-POWR1014 / A ispPAC-POWR1220AT8 Mature SVF	XCF File (.xcf) -Programmer Project file. XCF files can be used to save the setup of a Programmer Project. XCF files can either be loaded back into Programmer to work on a previous project or loaded into deployment tool for file conversions, generate a tester file, generate an embedded system file, or generate an external memory file.	VME File (.vme)

JTAG Full VME Embedded Options—Compress VME File, Convert VME Files to HEX (c.) File-Based Embedded VME

The following table lists options—Compress VME File, Convert VME Files to HEX (c.) File-Based Embedded VME—for JTAG Full VME embedded deployment.

Table 106:

Device Family	Compress VME		Convert VME Files to HEX (c.) File-Based Embedded VME	
ECP5	**On (Default)**	**Off**	**Off (Default)**	**On**
LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M Platform Manager Platform Manager 2 LatticeXP2 LatticeXP LFMNX MachXO3D MachXO3L MachXO2 MachXO ispXPGA ORCA ispMACH4000 / ZE ispPAC-CLK5300 ispPAC-CLK5400 ispPAC-POWR605 ispPAC-POWR607 ispPAC-POWR6AT6 ispPAC-POWR1014 / A ispPAC-POWR1220AT8 Mature SVF	Compress data. This option reduces the file size of the VME file through compression, using less memory space. Clearing this option generates an uncompressed VME, which is useful for debugging. Compressed and uncompressed VME have the same level of performance.	Uncompressed data.	By default this operation is turned off, this will create a VME file that will be used in file based embedded programming where the programming of the device is based on the input file. When this option is turned on then it will create a .c file which will be used in EPROM base programming to create a single complied image that will be loaded in to the CPU. An then the CPU will use this image to then program the device.	Generate Hex (.c) file.

JTAG Full VME Embedded Options—Compact VME File, Fixed Pulse Width (Rev. D)

The following table lists options—Compact VME File, Fixed Pulse Width (Rev. D)—for JTAG Full VME embedded deployment.

Table 107:

Device Family	Compact VME File		Fixed Pulse Width (Rev. D)	
Platform Manager	**Off (Default)**	**On**	**Off (Default)**	**On**
Platform Manager 2 LatticeXP2 LatticeXP LFMNX MachXO3D MachXO3L MachXO2 MachXOF	By default this operation is turned off. When this option is turned on it will split the VME data file into two different sections. An algorithm section and a data section. Whenever there is data needed in the algorithm section it will get it from the data section. This option is useful to turn on when the VME file is repeating the same operation even with different data. The operation will be in a loop and will reference the data section. This option will reduce the file size of the VME file.	Compact VME file.	By default this setting is off enabling looping (status polling algorithm) in the VME file. When it is turned on then there will be no looping (status polling algorithm) and instead commands in loops will be repeated for as many times as the loop iterated for. Status polling algorithm is a Lattice specific command in VME files, turning this option on makes the VME file generic and is useful when programming through a non-Lattice device, but increases file size.	Fixed Pulse Width VME file.

JTAG Full VME Embedded Options—Verify USERCODE, Program Device if Fails

The following table lists options—Verify USERCODE, Program Device if Fails—for JTAG Full VME embedded deployment.

Table 108:

Device Family	Verify USERCODE, Program Device if Fails	
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M Platform Manager Platform Manager 2 LatticeXP2 LatticeXP LFMNX MachXO3D MachXO3L MachXO2 MachXO ispXPGA ORCA ispMACH4000 / ZE ispPAC-CLK5300 ispPAC-CLK5400 ispPAC-POWR605 ispPAC-POWR607 ispPAC-POWR6AT6 ispPAC-POWR1014 / A ispPAC-POWR1220AT8 Mature	**Off (Default)**	**On**
	By default this operation is turn off and preforms the standard erase, program, verify flow. When this option is turned on then before attempting to program a device it will check to see if the USERCODE of the device is the same as the file to be programmed. If the USERCODEs are the same then that device will be skipped and will not be reprogrammed, otherwise if the USERCODEs do not match then the device will be programmed.	Verify USERCODE, Program device if USERCODE Verify Fails.

JTAG Full VME Embedded Options—Include Header, Include Comment, Maximum Memory Allocation Size Per Row of Data (Kbytes)

The following table lists options—Maximum Memory Allocation Size Per Row of Data (Kbytes)—for JTAG Full VME embedded deployment.

Table 109:

Device Family	Include Header		Include Comment		Maximum Memory Aloocation Size Per Row of Data (Kbytes)
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M Platform Manager Platform Manager 2 LatticeXP2 LatticeXP LFMNX MachXO3D MachXO3L MachXO2 MachXO ispXPGA ORCA ispMACH4000 / ZE ispPAC-CLK5300 ispPAC-CLK5400 ispPAC-POWR605 ispPAC-POWR607 ispPAC-POWR6AT6 ispPAC-POWR1014 / A ispPAC-POWR1220AT8 Mature SVF	**Off (Default)** By default the header is included in the VME file otherwise the header file is omitted from the VME file. The header contains information on the version and is useful in debugging. Turning the header off can be used to reduce the file size of the VME file size.	**On** Omit header from VME file.	**Off (Default)** The default is off which does not include comments in the VME file, if turned on then comments will appear for the operations. Turning comments on is useful for debugging the VME file, but will increase file size.	**On** Include comments in VME file.	Set the Maximum buffer size. By default this size of the buffer is 64k, but if a system has a limitation on the max buffer size only being 16k then this option can be set to 16k to compensate for that limitation. Or if the system can support buffers of more than 64k than the setting can be increase to 128k or 256k. Valid values for this operation are 8, 16, 32, 64, 128, 256.

JTAG Slim VME Embedded—Device Family, Input File 1, Output File 1, and Output File 2

The following table lists device family, input file types, and output file. extensions for JTAG Slim VME embedded deployment.

Table 110:

Device Family	Input File	Output File 1	Output File 2
Lattice Chain	XCF File (.xcf)	Algorithm VME File (*_algo.vme)—This is the algorithm file that specifies which operations will be executed must be used with a VME data file. This file is used in device programming or in field upgrading.	Data VME File (*_data.vme)—This is the data file that provides the data that the algorithm file will use to preforms executed operations must be used with a VME algorithm file. This file is used in device programming or in field upgrading.

JTAG Slim VME Embedded Options—Compressed VME Data File, Convert VME files to HEX (c.) File-Based Embedded VME

The following table lists options—Compressed Embedded Files, onvert VME files to HEX (c.) File-Based Embedded VME—for JTAG Slim VME embedded deployment.

Table 111:

Device Family	Compress VME Data File		Convert VME files to HEX (c.) File-Based Embedded VME	
Lattice	**On (Default)**	**Off**	**Off (Default)**	**On**
	Compress data. This option reduces the file size of the VME file through compression, using less memory space. Clearing this option generates an uncompressed VME, which is useful for debugging. Compressed and uncompressed VME have the same level of performance.	Do not compress.	By default this operation is turned off, this will create a VME file that will be used in file based embedded programming where the programming of the device is based on the input file. When this option is turned on then it will create a .c file which will be used in EPROM base programming to create a single complied image that will be loaded in to the CPU. An then the CPU will use this image to then program the device.	Generate Hex (.c) file.

Slave SPI Embedded—Device Family, Input File 1, Output File 1, and Output File 2

The following table lists device family, input file types, and output file extensions for Slave SPI embedded deployment.

Table 112:

Device Family	Input File	Output File 1	Output File 2
LFMNX MachXO3D ECP5 LatticeECP3 LIFMD LIFMDF	XCF File (.xcf) Bitstream (.bit)	Algorithm VME File (*_algo.vme)—This is the algorithm file that specifies which operations will be executed must be used with a VME data file. This file is used in device programming or in field upgrading.	Data File (*_data.sed)
LatticeXP2	XCF File (.xcf) JEDEC (.jed)		

Slave SPI Embedded Options—Compress Embedded Files, Convert VME files to HEX (c.) File-Based Embedded VME

The following table lists options—Compress Embedded Files, Convert VME files to HEX (c.) File-Based Embedded VME—for Slave SPI embedded deployment.

Table 113:

Device Family	Convert VME files to HEX (c.) File-Based Embedded VME		Generate Hex (.c) File		Operation (.xcf and .jed files only)
MachXO3D ECP5 LatticeECP3 LatticeXP2 LFMNX LIFMD LIFMDF	**On (Default)** By default this operation is turned off, this will create a VME file that will be used in file based embedded programming where the programming of the device is based on the input file. When this option is turned on then it will create a .c file which will be used in EPROM base programming to create a single complied image that will be loaded in to the CPU. An then the CPU will use this image to then program the device.	**Off** Do not compress	**Off (Default)** Do not generate Hex (.c) file	**On** Generate Hex (.c) file.	The list of options and operation is dependent on which device is selected.

I2C Embedded—Device Family, Input File 1, Output File 1, and Output File 2

The following table lists device family, input file types, and output file extensions for I2C embedded deployment.

Table 114:

Device Family	Input File	Output File 1	Output File 2
LFMNX MachXO2 MachXO3D MachXO3L Platform Manager 2	XCF File (.xcf) JEDEC (.jed)	Algorithm File (*_algo.sea)	Data File (*_data.sed)

I2C Embedded Options—Compress Embedded Files, Convert VME files to HEX (c.) File-Based Embedded VME, Operation (.xcf and .jed files only), I2C Slave Address

The following table lists options—Compress Embedded Files, Convert VME files to HEX (c.) File-Based Embedded VME, Operation (.xcf and .jed files only), I2C Slave Address—for I2C embedded deployment.

Table 115:

Device Family	Compressed Embedded Files		Convert VME files to HEX (c.) File-Based Embedded VME		Operation (.jed files only)	I2C Slave Address
LFMNX LIFMD LIFMDF MachXO2 MachXO3D MachXO3L Platform Manager 2	**On (Default)**	**Off**	**Off (Default)**	**On**	The list of options and operation is dependent on which device is selected.	Used to set the I2C Slave address. By default the value is 0b1000000. The user can modify this value if the value of the I2C Slave Address if there project calls for it.
	Compress embedded files.	Do not compress.	By default this operation is turned off, this will create a VME file that will be used in file based embedded programming where the programming of the device is based on the input file. When this option is turned on then it will create a .c file which will be used in EPROM base programming to create a single complied image that will be loaded in to the CPU. An then the CPU will use this image to then program the device.	Generate Hex (.c) file.		

I2C Embedded Options—Include Comments

The following table lists options—Include Comments for I2C embedded deployment.

Table 116:

Device Family	Include Comments	
LFMNX	**Off (Default)**	**On**
LIFMD LIFMDF MachXO2 MachXO3D MachXO3L Platform Manager 2	The default is off which does not include comments in the VME file, if turned on then comments will appear for the operations. Turning comments on is useful for debugging the VME file, but will increase file size.	Include comments.

I2C Embedded Options—Fixed Pulse Width

The following table lists options—Fixed Pulse Width for I2C embedded deployment.

Table 117:

Device Family	Fixed Pulse Width	
LFMNX	**Off (Default)**	**On**
MachXO2 MachXO3D MachXO3L Platform Manager 2	By default this setting is off enabling looping (status polling algorithm) in the VME file. When it is turned on then there will be no looping (status polling algorithm) and instead commands in loops will be repeated for as many times as the loop iterated for. Status polling algorithm is a Lattice specific command in VME files, turning this option on makes the VME file generic and is useful when programming through a non-Lattice device, but increases file size.	Used fixed pulse width.

sysCONFIG (CPU) Embedded—Device Family, Input File 1, Format, and Output File 1

The following table lists device family, input file types, and output file extensions for I2C embedded deployment

Table 118:

Device Family	Input File 1	Format	Output File
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeXP ispXPGA	XCF File (.xcf)	Binary	Embedded Bitstream (.cpu)
		C-Code	Embedded Bitstream (.c)
		Intel Hex	Embedded Bitstream (.hex)
		Text (Debug Only)	Embedded Bitstream (.txt)

sysCONFIG (CPU) Embedded—Options—For Erase, Program, and Verify Operations, Skip Verify; Include Comments

The following table lists options—For Erase, Program, and Verify Operations, Skip Verify; Include Comments—for sysCONFIG (CPU) embedded deployment.

Table 119:

Device Family	For Erase, Program, and Verify Operations, Skip Verify		Include Comments	
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeXP ispXPGA	**On (Default)**	**Off**	**Off (Default)**	**On**
	By default this option is turned on, leave this option on if the user wishes to not verify the CPU file programmed in. By turning this option off this will perform the Erase, Program, Verify operations which will reprogram the device and verify the contents of the device. If the user wants to verify that the Contents of CPU file they should turn this option off.	Include verify.	Include comments in CPU file.	Do not include comments in CPU file.

sysCONFIG (CPU) Embedded—Options—Compress Embedded Files, Byte Wide Bit Mirror

The following table lists options—Compress Embedded Files, Byte Wide Bit Mirror—for sysCONFIG (CPU) embedded deployment.

Table 120:

Device Family	Compress Embedded Files (Binary, C-code, and Intel Hex only)		Byte Wide Bit Mirror (Intel Hex and Text—Debug Only)	
ECP5	**On (Default)**	**Off**	**Off (Default)**	**On**
LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeXP ispXPGA	By default the this operation is off. When this operation is off it includes comments in the CPU file, this is useful for debugging to see which operations are being performed. Turning this option off cause the CPU file to not include the comments. This is useful in reducing the file size if there is a memory constraint.	Do not compress.	By default this is turned off. When this is turned on each byte of data will be flipped. This can be used if the pin mapping on the device is the mirror of the Lattice devices, instead of remapping the pins the user can just select this option.	Flip each byte of data.

See Also ▶"Input File Descriptions" on page 993

External Memory Deployment Type

This topic provides tables that list device family, input file types, output file extensions, and options for File Conversion deployment function type.

Hex Conversion

The following table lists device family, input file types, and output file extensions for each External Memory file type.

Table 121:

Device Family	Input File	Output File 1	Output File 2
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M ORCA Unknown or Non-Lattice	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tektronix Hex	Hex (.xtek)
LatticeXP2 LatticeXP LFMNX LIFMD LIFMDF MachXO2 MachXO3D MachXO3L	JEDEC (.jed) Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tektronix Hex	Hex (.xtek)

Hex Conversion Options—Program Security Bit

The following table lists options—Program Security Bit—for Hex conversion deployment.

Table 122:

Device Family	Program Security Bit		
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticECP LatticeXP2 LatticeXP LIFMD LIFMDF	**Default (Default)**	**Off**	**On**
	Use setting from input file.	Do not program security fuses.	Program security fuses.

Hex Conversion Options—Encryption

The following table lists options—Encryption—for Hex conversion deployment.

Table 123:

Device Family	Encryption	
	Configuration Mode	**Encryption Key**
		Load from BEK File
LFMNX MachXO3D ECP5 LatticeECP3 LatticeECP2S/MS	Selects the appropriate configuration mode for the bitstream encryption.	Load from BEK file.

Hex Conversion Options—Authentication

The following table lists options—Encryption—for Hex conversion deployment.

Table 124:

Device Family	Authentication
LFMNX MachXO3D	**Public Key and Private Key or Signature Key**
	Load from GUI and load Public Key file (.pub) and Private File (.prv).

Hex Conversion Options—Verify ID Code, Frequency, Compression

The following table lists options—Verify ID Code, Frequency, and Compression—for hex conversion deployment.

Figure 11:

Device Family	Verify ID Code			Frequency		Compression		
ECP5 LFMNX	**Default (Default)**	**On**	**Off**	**Default (Default)**	**Frequency Selection**	**Default (Default)**	**On**	**Off**
MachXO2 MachXO3D MachXO3L	Use setting from input file.	Include verify ID in bitstream .	Do not verify ID code.	Use setting from input file.	Valid frequency device dependent.	Use setting from input file.	Compress bistream data.	Do not compress .

Figure 11:

Device Family	Verify ID Code			Frequency		Compression		
LatticeECP3, LatticeXP, LatticeXP2	Use setting from input file.	Include verify ID in bitstream .	Do not verify ID code.	Use setting from input file.	Valid frequency device dependent.	-	-	-
LatticeSC/M	-	-	-	-	-	Use setting from input file.	Compress bistream data.	Do not compress .

Hex Conversion Options—CRC Calculation

The following table lists options—CRC Calculation—for hex conversion deployment.

Figure 12:

Device Family	CRC Calculation		
	Default (Default)	**On**	**Global CRC Only**
ECP5 LatticeECP2S/MS LatticeECP2/M LatticeXP2 LatticeXP LFMNX MachXO2 MachXO3D MachXO3L	Use setting from input file.	Include CRC for each Data Frame.	Include Global CRC Only.

Hex Conversion Options—Byte Wide Mirror

The following table lists options—Byte Wide Mirror—for hex conversion deployment.

Figure 13:

Device Family	Off (Default)	On
ECP5	No change to bit order.	No change to bit order.
LatticeECP3		
LatticeECP2S/MS		
LatticeECP2/M		
LatticeEC/P		
LatticeSC/M		
LatticeXP2		
LatticeXP		
LFMNX		
MachXO2		
MachXO3D		
MachXO3L		
ORCA		
Unknown or Non-Lattice		

Hex Conversion Options—Retain Bitstream Header

The following table lists options—Retain Bitstream Header—for hex conversion deployment.

Figure 14:

Device Family	Off (Default)	On
ECP5	Replace Bitstream Header.	Retain Bitstream Header.
LatticeECP3		
LatticeEC/P		
LatticeSC/M		
LatticeXP2		
LFMNX		
LatticeXP		
MachXO2		
MachXO3D		
MachXO3L		
ORCA		

Hex Conversion Options—Starting Address

The following table lists options—Starting Address—for hex conversion deployment.

Figure 15:

Device Family	Starting Address	
LatticeECP3	**Off (Default)**	**Starting Address**
LatticeECP2S/MS	No address offset.	Starting address in 0x010000 increments.
LatticeECP2/M		
LatticeEC/P		
LatticeSC/M		
Platform Manager		
Platform Manager 2		
LatticeXP2		
LatticeXP		
LFMNX		
MachXO3D		
MachXO3L		
MachXO2		
MachXO		
ispXPGA		
ORCA		
ispMACH4000 / ZE		
ispPAC-CLK5300		
ispPAC-CLK5400		
ispPAC-POWR605		
ispPAC-POWR607		
ispPAC-POWR6AT6		
ispPAC-POWR1014 / A		
ispPAC-POWR1220AT8		
Mature		

Hex Conversion Options—SPI Flash Read Mode and Starting Address (ECP5 only)

The following table lists options—SPI Flash Read Mode and Starting Address—for hex conversion deployment.

Figure 16:

Device Family	SPI Flash Mode		Starting Address	
ECP5	**Off (Default)**	**Fast Read**	**Off (Default)**	**Starting Address**
	Standard Read Opcode.	Fast Read Opcode.	No address offset.	Starting address in 0x010000 increments.

Dual Boot

The following table lists device family, input file types, format, and output file extensions for dual boot deployment

Table 125:

Device Family	Golden Pattern Input File 1	Primary Pattern Input File 2	Format	Output File
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LIFMD LIFMDF	Bitstream (.bit, .rbt)	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
			Motorola Hex	Hex (.exo)
			Extended Tektronix Hex	Hex (.xtek)
MachXO2 Platform Manager 2	ASC File(s) (.hex)	-	Intel Hex	Hex (.mcs)
			Motorola Hex	Hex (.exo)
			Extended Tektronix Hex	Hex (.xtek)
LFMNX MachXO3L MachXO3D		-	Intel Hex	Hex (.mcs)
			Motorola Hex	Hex (.exo)
			Extended Tektronix Hex	Hex (.xtek)

Dual Boot Options—SPI Flash Size

The following table lists options—SPI Flash Size—for dual boot deployment.

Figure 17:

Device Family	SPI Flash Size
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LFMNX LIFMD LIFMDF MachXO2 MachXO3D MachXO3L Platform Manager 2	1/2/4/8/16/32/64/128/256/513 (512 Mb)

Dual Boot Options—Protect Golden Sector

The following table lists options—Protect Golden Sector—for dual boot deployment.

Table 126:

Device Family	Protect Golden Sector	
MachXO3D	**Off (Default)**	**On**
ECP5 LFMNX LatticeECP3 LatticeECP2S/MS LatticeECP2/M	Locate golden in next available sector.	Locate Golden at the beginning of the upper half.

Dual Boot Options—Bye Wide Mirror and Retain Bitsteam Header

The following table lists options—Bye Wide Mirror and Retain Bitsteam Header—for dual boot deployment.

Figure 18:

Device Family	Byte Wide Mirror		Retain Bitstream Header	
ECP5	**Off (Default)**	**On**	**Off (Default)**	**On**
LatticeECP3 LatticeECP2S/MS LatticeECP2/M LFMNX MachXO2 MachXO3D MachXO3L Platform Manager 2	No change to bit order.	Flip each byte of data.	Replace Bitstream Header.	Retain Bitstream Header.

Dual Boot Options—External ASC Devices

The following table lists options—External ASC Devices—for dual boot deployment.

Figure 19:

Device Family	External ASC Devices					
	Off (Default)	**On**				
			Number of External ASC Devices	**Select External ASC Data File 1**	**External ASC 2-6**	**Select Last External ASC Data File**
MachXO2	No External ASC.	External ASC.	1/2/3/4/5/6/7/8	Select External ASC Data File 1.	External ASC 2-7.	Select External ASC Data File 8.
Platform Manager 2	No External ASC.	External ASC.	1/2/3/4/5/6/7	Select External ASC Data File 1.	External ASC 2-6.	Select External ASC Data File 7.

Dual Boot Options—SPI Flash Read Mode

The following table lists options—SPI Flash Read Mode—for dual boot deployment.

Figure 20:

Device Family	SPI Flash Mode			
ECP5	**Off (Default)**	**Fast Read**	**Dual I/O**	**Quad I/O**
	Standard Read Opcode.	Fast Read Opcode.	Dual I/O Read Opcode.	Quad I/O Read Opcode.

Dual Boot Options—Optimize Memory Space

The following table lists options—Optimize Memory Space—for dual boot deployment.

Figure 21:

Device Family	SPI Flash Mode	
	Off (Default)	**On**
MachXO3D	Use worst case bitstream size.	Use file size.
ECP5		
LatticeECP3		
LatticeECP2S/MS		
LatticeECP2/M		
LFMNX		
LIFMD		
LIFMDF		

Dual Boot Options—Encryption

The following table lists options—Encryption—for dual boot deployment..

Table 127:

Device Family	Off (Default)	On
ECP5	Normal bitstream.	Primary encrypted bitstream.
LatticeECP3		
LatticeECP2/MS		
LatticeECP2/M		
LatticeECP		
LFMNX		
MachXO3D		

Table 127:

Device Family	Off (Default)	On
LatticeXP2 LatticeXP MachXO2	Binary Bitstream	Bitstream (.bit)
	ASCII Bitstream	Bitstream (.rbt)
	Intel Hex	Hex (.mcs)
	Motorola Hex	Hex (.exo)
	Extended Tektronix Hex	Hex (.xtek)

Dual Boot Options—Authentication

The following table lists options—Authentication—for dual boot deployment.

Figure 22:

Device Family	Off (Default)	On
LFMNX MachXO3D	Normal bitstream.	Primary authentication bitstream.

Advanced SPI Flash

The following table lists device family, input file types, format, and output file extensions for external SPI Flash deployment.

Table 128:

Device Family	Input File 1	Format	Output File
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M LFMNX LIFMD LIFMDF MachXO2 MachXO3D MachXO3L	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
		Motorola Hex	Hex (.exo)
		Extended Tektronix Hex	Hex (.xtek)

Advanced SPI Flash Options—SPI Flash Size

The following table lists options—SPI Flash Size—for advanced SPI Flash deployment.

Figure 23:

Device Family	SPI Flash Size
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M LFMNX MachXO2 MachXO3D MachXO3L	512 (512Kb)/1/2/4/8/ 16/32/64/128/256/ 513 (512Mb)

Advanced SPI Flash Options—Bye Wide Mirror

The following table lists options—Bye Wide Mirror and Retain Bitsteam Header—for advanced SPI Flash deployment.

Figure 24:

Device Family	Byte Wide Mirror	
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M MachXO2 MachXO3L	**Off (Default)**	**On**
	No change to bit order.	Flip each byte of data.

Advanced SPI Flash Options—Retain Bitstream Header

The following table lists options—Bye Wide Mirror and Retain Bitsteam Header—for advanced SPI Flash deployment.

Figure 25:

Device Family	Retain Bitstream Header	
ECP5	**Off (Default)**	**On**
LatticeECP3	Replace Bitstream Header.	Retain Bitstream Header.
LatticeECP2S/MS		
LatticeECP2/M		
LatticeEC/P		
LatticeSC/M		
LFMNX		
MachXO2		
MachXO3D		
MachXO3L		

Advanced SPI Flash Options—User Hex Files

The following table lists options—User Hex Files—for advanced SPI Flash deployment.

Figure 26:

Device Family								
ECP5 LatticeECP3	**Off (Default)**	**On**						
LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeSC/M LFMNX MachXO2 MachXO3D MachXO3L	Off	User data	**Number of User Hex Files**	**Select User Hex File 1**	**Select Starting Address 1**	**User Data 2-3**	**Select User Hex File 4**	**Select Starting Address 4**
			1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16	Select User Hex File	0x010000 Sectors	User Data 2 - 3	Select User Hex File	0x010000 Sectors

Advanced SPI Flash Options—SPI Flash Read Mode

The following table lists options—SPI Flash Read Mode—for advanced SPI Flash deployment.

Figure 27:

Device Family				
ECP5 MachXO2 MachXO3D MachXO3L LFMNX LIFMDF	**Off (Default)**	**Fast Reset**	**Dual I/O**	**Quad I/O**
	Off	Fast Read Opcode.	Dual I/O Read Opcode.	Quad I/O Read Opcode.

Advanced SPI Flash Options—Multiple Boot

The following table lists options—SPI Flash Read Mode—for advanced SPI Flash deployment.

Figure 28:

Device Family	Multiple Boot												
ECP5 LFMNX MachXO2 MachXO3D MachXO3L	**Off (Default)**	**On**										**Optimize Memory Space**	
	No Multiboot	Multiboot	**Number of Alternate Data Files**	**Select Golden Data File**	**Select Alternate Data File 1**	**Select Starting Address 1**	**Next Pattern**	**Alternates 2 - 3**	**Select Alternate Data File 4**	**Select Starting Address 4**	**Next Pattern**	**Off (Default)**	**On**
			1/2/3/4	Select Golden Data File.	Select Alternate Data File.	0x010000 Sectors.	<prim|alt1|alt2|alt3|alt4>.	Alternates 2 - 3.	Select Alternate Data File.	0x010000 Sectors.	<prim|alt1|alt2|alt3|alt4>.	Use worst case bitstream size.	Use file size.

sysCONFIG Daisy Chain

The following table lists device family, input file types, format, and output file extensions for sysCONFIG daisy chain deployment

Table 129:

Device Family	Input File 1	Input File 2	Format	Output File
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeEC/P LatticeXP LatticeSC/M ORCA Unknown or Non-Lattice	Bitstream (.bit, .rbt)	Bitstream (.bit, .rbt)	Intel Hex	Hex (.mcs)
			Motorola Hex	Hex (.exo)
			Extended Tektronix Hex	Hex (.xtek)

sysCONFIG Daisy Chain Options—Merge Format

The following table lists options—Merge Format—for sysCONFIG daisy chain deployment.

Table 130:

Device Family	Merge Format			
	Intelligent (Default)	Intelligent Merge		Combine
		Frequency	Slayer Output	
ECP5 LatticeECP3 LatticeECP2S/MS LatticeECP2/M LatticeECP	Intelligent Merge.	Default/(Device Specific Selections).	-	Combine Merge.
Lattice XP	Intelligent Merge.	-	-	Combine Merge.
LatticeSC/M ORCA	Intelligent Merge.	-	LatticeSC/M Output Pin.	Combine Merge.
Unknown or Non-Lattice	-	-	-	Combine Merge.

sysCONFIG Daisy Chain—Options—Byte Wide Bit Mirror and Retain Bitstream Header

The following table lists options—Byte Wide Bit Mirror and Retain Bitstream Header—for sysCONFIG Daisy Chain deployment.

Table 131:

Device Family	Byte Wide Bit Mirror		Retain Bitstream Header	
ECP5	**Off (Default)**	**On**	**Off (Default)**	**On**
LatticeECP3 LatticeEC/P LatticeXP LatticeSC/M ORCA	No change to bit order.	Flip each byte of data.	Replace Bitstream Header.	Retain Bitstream Header.
LatticeECP2S/MS LatticeECP2/M Unknown or Non-Lattice	No change to bit order.	Flip each byte of data.	-	-

See Also ▶Input File Descriptions

Input File Descriptions

The following table shows how files produced by Diamond (and ispLEVER for mature devices) and the Deployment Tool can be used as input files to generate other data file formats. For example, the Deployment Tool uses a binary bitstream generated by Diamond for a LatticeEC device to output an ISC file. This ISC file can then be used to generate a bitstream ASCII readback file.

The Deployment Tool can read customized bitstreams with various commands and options, such as CRC calculation and comparison.

Table 132:

Input File	Description	Output
.jed	Joint Electronic Design Engineering Council standard data file (JEDEC) generated from an ISC, BIT, or RBT	Bitstream, SVF, ISC, STAPL, Intel, Motorola, and Extended Tektronix Hex
.isc	In-System Configurable file (ISC) generated from a JEDEC, BIT, RBT, RBK, RBKA, MSK, or MSKA	Bitstream, SVF, JED, STAPL
.bit	Bitstream binary file (BIT) generated from a JEDEC (ispXPLD devices only), ISC, BIT, or RBT	Bitstream, ISC, JED, Intel, Motorola, and Extended Tektronix Hex

Table 132:

Input File	Description	Output
.rbt	Bitstream ASCII file (RBT) generated from a JEDEC (ispXPLD devices only), ISC, BIT, or RBT	Bitstream, ISC, JED, Intel, Motorola, and Extended Tektronix Hex
.rbk	Bitstream binary readback file (RBK) generated from an ISC when the READ option is selected for bitstream binary output.	ISC
.rbka	Bitstream ASCII readback file (RBKA) generated from an ISC when the READ option is selected for bitstream ASCII output.	ISC
.xcf	An .xcf file is a configuration file used by the Diamond Programmer and by ispVM System for programming devices in a JTAG daisy chain. The XCF file contains information about each device, the data files targeted, and the operations to be performed.	STAPL File (.stp), SVF File (.svf), Generic Vector Format (.tst), GenRad (.gr), HP3070 (.pcf), HP3065 (.pcf), Teradyne 1800 (.asc), Teradyne L200/ 300 (.asc), ME File (.vme), Data VME File (*_data.vme), Data File (*_data.sed), Embedded Bitstream (.cpu), Embedded Bitstream (.c), Embedded Bitstream (.txt), Embedded Bitstream (.hex)

See Also ▶"Input File Formats" on page 994

Input File Formats

The input file format used with the Deployment Tool depends on the device and the selected output format. The following table shows the data files for Lattice devices that can be used as input files to generate bitstream, serial vector format, or in-system configuration data files.

Table 133:

Input for Bitstream Output	Multiple Input/ Output	Input for SVF Output	Input for ISC Output	Input for JEDEC Output
LatticeECP/EC, LatticeECP2, LatticeECP3, LatticeSCM/SC, ECP5U/UM				
.bit, .rbt	.bit, .rbt	.bit, .rbt	Not available	Not available
LatticeXP				
.bit, .isc, .jed, .rbt	Not available	.jed	Not available	.bit, .rbt, .jed
MachXO, MachXO2				
Not available	Not available	.jed	Not available	Not available

Table 133:

Input for Bitstream Output	Multiple Input/ Output	Input for SVF Output	Input for ISC Output	Input for JEDEC Output
ORCA Series 2, 3, 4, ORCA FPSC				
.bit, .rbt	Not available	.bit, .rbt	Not available	Not available
ispXPGA				
.jed, .isc	.isc	.jed, .isc	.jed	.isc, .jed
ispXPLD				
.jed, .isc	.jed	.jed, .isc	.jed	.isc
CPLD				
Not available	Not available	.jed, .isc	.jed	.isc, .jed

See Also ▶"Input File Descriptions" on page 993

Bitstream Output File Descriptions

The output file format generated by the Deployment Tool depends on the device and the selected format options. To see a list of all the output formats by device family, see "Bitstream Output Formats" on page 996.

Table 134:

File Extension	Description
.bit	Binary bitstream.
.rbt	ASCII bitstream.
.msk	Binary bitstream mask file generated from an In-System Configurable (ISC) file when the READ_MASK option is selected for bitstream binary output.
.mska	ASCII bitstream mask file generated from an In-System Configurable (ISC) file when the READ_MASK option is selected for ASCII output.
.rbk	Binary readback bitstream file generated from an ISC file when the READ option is selected for binary output.
.rbka	ASCII readback bitstream file generated from an ISC file when the READ option is selected for ASCII output.
.exo	A Motorola hexadecimal output data file.
.mcs	An Intel hexadecimal output data file.
.xtek	An Extended Tektronix hexadecimal output data file.

Bitstream Output Formats

The Deployment Tool generates bitstream data files for the following Lattice device families: LatticeECP/EC, LatticeECP2 LatticeECP3, LatticeXP, and LatticeSCM/SC; ORCA Series 2, 3, 4; ORCA FPSC; ispXPGA; ispXPLD. The table below shows the types of bitstreams and output formats that can be generated for each of these families.

Table 135:

ECP5 LatticeECP/EC, LatticeECP2, LatticeECP3, LatticeXP, LatticeSCM/SC	**ORCA2, 3, 4, ORCA FPSC**	**ispXPGA**	**ispXPLD**
Single Bitstream			
Bitstream Binary (.bit) Bitstream ASCII (.rbt) Intel Hex (.mcs) Motorola Hex (.exo) Extended Tektronix Hex (.xtek)	Intel Hex (.mcs) Motorola Hex (.exo) Extended Tektronix Hex (.xtek)	PCM Binary (.pcm) PCM Hex (.pcm) Intel Hex (.mcs) Motorola Hex (.exo) Extended Tektronix Hex (.xtek)	PCM Binary (.pcm) PCM Hex (.pcm) Intel Hex (.mcs) Motorola Hex (.exo) Extended Tektronix Hex (.xtek)
Merged Bitstream			
Intel Hex Merge (.mcs) Motorola Hex Merge (.exo) Extended Tektronix Hex Merge (.xtek)	Intel Hex (.mcs) Motorola Hex (.exo) Extended Tektronix Hex Merge (.xtek)	Unavailable	Unavailable

See Also

- “Bitstream Output File Descriptions” on page 995
- “Input File Descriptions” on page 993
- “Output Options Descriptions” on page 997

SVF, ISC, JEDEC, and STAPL Output Formats

The Deployment Tool generates Serial Vector Format (SVF), In-System Configuration (ISC), Joint Electronic Design Engineering Council (JEDEC), and Standard Test and Programming Language (STAPL) data files for Lattice devices.

SVF Output Files

SVF files are generated from JEDEC, ISC, binary bitstream (BIT), or ASCII bitstream (RBT) input files.

ISC Output Files

ISC files are generated from JEDEC, binary bitstream (BIT), ASCII bitstream (RBT), binary readback (RBK), ASCII readback (RBKA), binary mask (MSK), and ASCII mask (MSKA) input files.

JEDEC Output Files

JEDEC files are generated from ISC, BIT, and RBT input files.

STAPL Output Files

STAPL files are generated from JEDEC input files.

See Also ▶"Input File Descriptions" on page 993

Output Options Descriptions

The following describes each of the options provided by the Deployment Tool for outputting data files. The number and types of options depend on the type of device that you are programming and the output format. For options available by device type, refer to the device-specific options.

Verify ID Code – Generates the bitstream which verifies the ID code.

Program Secure – Generates the data file which secures the device.

Compression – Compresses the bitstream.

Frequency – Generates the bitstream using the specified frequency.

Byte Wide Bit Mirroring – Flips each byte in Intel, Extended Tektronix, and Motorola hexadecimal data files.

RUNTEST from REV C – Generates the serial vector format file using the format specified in the SVF file specification, revision C or earlier.

For Erase, Program, and Verify Operations, Skip Verify – Globally skips the verification step of the Erase, Program, and Verify operation.

Creating a New Deployment

Deployment Tool has a wizard interface that allows you step through the process of creating a deployment for the various function types and output file types.

To create a new Deployment:

1. Issue the start command.
   - In Linux, from the from the <*Programmer install path*>/bin/lin directory, enter the following on a command line:

     ```
     ./deployment
     ```
2. In the Getting Started dialog box, choose **Create a New Deployment.**
3. In the Function Type dropdown menu, choose from one of the following options:
   - File Conversion
   - Tester
   - Embedded System
   - External Memory
4. In the Output File Type box, choose an output file type.
5. Click **OK**. The Deployment Tool loads the device database.
6. If you are creating a deployment from an .xcf file, in the Step 1 of 4 dialog box, use the ... **(Browse)** button to browse to the project's .xcf file.

   If you are creating a deployment from a data file, in the Step 1 of 4 dialog box, select Input File(s) window, click **... (Browse)** in the File Name box. The Device Family and Device boxes are automatically populated by the Deployment Tool.
7. Click **Next**.
8. In the Step 2 of 4 Options dialog box, select the desired options for your output file type.
9. Click **Next**.
10. In the Step 3 of 4 Select Output File(s) dialog box, select the desired output file type from the dropdown menu, and using the using the ... button, browse to the desired location of the output file(s).
11. Click **Next**.
12. In the Step 4 of 4 Generate Deployment dialog box, review Deployment Tool Summary and Command Line Information.
13. Choose **File > Generate**, or click the  button, to perform the Deployment. The Deployment Generate Result are available for review.
14. If you wish to save the deployment, choose **File > Save**, or click the  button, to save the deployment (.ddt) file in your desired location.

See Also ▶"Input File Descriptions" on page 993

- "Output Options Descriptions" on page 997
- "Bitstream Output Formats" on page 996
- "Bitstream Output File Descriptions" on page 995

Opening an Existing Deployment

If a deployment was saved as a deployment (.ddt) file, the deployment can be opened in Deployment Tool and regenerated using the same deployment options, or by applying different deployment options.

To open an existing deployment:

1. Issue the start command.
   - In Linux, from the from the <*install path*>/bin/lin directory, enter the following on a command line:

     ```
     ./deployment
     ```
2. In the Getting Started dialog box, choose **Open an Existing Deployment.**
3. Click **OK**. The Deployment Tool loads the device database.
4. If you are creating a deployment from a .ddt file, in the Step 1. of 4 dialog box, using the ... button, browse to the and select the project's .ddt file.

   If you are creating a deployment from a data file, in the Step 1. of 4 dialog box, select Input File(s) window, click **... (Browse)** in the File Name box. The Device Family and Device boxes are automatically populated by the Deployment Tool.
5. Click **Next**.
6. In the Step 2 of 4 Options dialog box, select the desired options for your output file type.
7. Click **Next**.
8. In the Step 3 of 4 Select Output File(s) dialog box, select the desired output file type from the dropdown menu, and using the using the ... button, browse to the desired location of the output file(s).
9. Click **Next**.
10. In the Step 4 of 4 Generate Deployment dialog box, review Deployment Tool Summary and Command Line Information.
11. Choose **File > Generate**, or click the  button, to perform the Deployment. The Deployment Generate Result are available for review.
12. If you wish to save the deployment, choose **File > Save**, or click the  button, to save the deployment (.ddt) file in your desired location.

See Also ▶"Creating a New Deployment" on page 997

- "Input File Descriptions" on page 993
- "Output Options Descriptions" on page 997
- "Bitstream Output Formats" on page 996
- "Bitstream Output File Descriptions" on page 995

Encrypting a JEDEC with Deployment Tool

Encryption settings are initially set in the Diamond software using the Security Setting Tool. Refer to "Setting Security Options" on page 76.

Deployment Tool allows you to encrypt an existing unencrypted JEDEC (.jed) file it has not already been encrypted, and output an encrypted .jed file.

Note

LFMNX, LatticeXP2 family, and MachXO3D family support encrypted JEDEC files.

IMPORTANT!

To turn on this security feature, you need to install the Encryption Control Pack.

To encrypt an unencrypted .jed file with Deployment Tool:

1. Issue the start command.
   - In Linux, from the <*install_path*>/bin/lin directory, enter the following on a command line:

     ```
     ./deployment
     ```
2. In the Getting Started dialog box, choose **Create a New Deployment.**
3. In the Function Type dropdown menu, choose **File Conversion**.
4. In the Output File Type box, choose **JEDEC File**.
5. Click **OK**. The Deployment Tool loads the device database.
6. In the Step 1 of 4 dialog box, select Input File(s) window, click **... (Browse)** in the File Name box, and browse to and open the .bit, .isc., .jed, or .rbt file you wish to encrypt, click **Open**, then click **Next**.
7. In the Step 2 of 4 Options dialog box,click the **Encryption** box.
8. To edit the encryption key, click **Edit Key** to display the Encryption Key dialog box. Edit the encryption key as desired, or load an Encryption Key Format file (.bek or .BEK) from file, and click **OK**.

   Note

   If the encryption key was entered manually, you must save the key to a .BEK file.
9. Click **Next**.
10. In the Step 3 of 4 Select Output File(s) dialog box, select the desired output file type from the dropdown menu, and using the ... button, browse to the desired location of the output file(s).
11. Click **Next**.

12. In the Step 4 of 4 Generate Deployment dialog box, review Deployment Tool Summary and Command Line Information.
13. Choose **File > Generate**, or click the button, to perform the Deployment. The Deployment Generate Result are available for review.
14. If you wish to save the deployment, choose **File > Save**, or click the button, to save the deployment (.ddt) file in your desired location.

For more information about setting security options for Lattice FPGA devices, refer to the following Lattice Technical Notes:

- TN1212, LatticeXP2 Advanced Security Programming Usage Guide
- TN1215, Advanced Security Encryption Key Programming Guide for ECP5, LatticeECP3, and LatticeECP2/MS Devices

See Also ▶"Setting Security Options" on page 76

- "Encrypting a Bitstream with Deployment Tool" on page 1001

Encrypting a Bitstream with Deployment Tool

Encryption settings are initially set in the the Diamond software using the Security Setting Tool. Refer to "Setting Security Options" on page 76.

Deployment Tool allows you to encrypt an existing unencrypted bitstream (.bit or .rbt) file if it was not already encrypted, and output an encrypted .bit file.

Note

Only the ECP5, LatticeECP2S/MS, MachXO3D, LFMNX, and LatticeECP3 devices support encrypted bitstreams.

IMPORTANT!

To turn on this security feature, you need to install the Encryption Security Control Pack.

To encrypt an unencrypted .bit or .rbt file with Deployment Tool:

1. Issue the start command.
   - In Linux, from the <*install_path*>/bin/lin directory, enter the following on a command line:

     ```
     ./deployment
     ```
2. In the Getting Started dialog box, choose **Create a New Deployment.**
3. In the Function Type dropdown menu, choose **File Conversion**.
4. In the Output File Type box, choose **Bitstream**.

5. Click **OK**. The Deployment Tool loads the device database.
6. In the Step 1 of 4 dialog box, select Input File(s) window, click **... (Browse)** in the File Name box, and browse to and open the .bit you wish to encrypt, then click **Next**.
7. In the Step 2 of 4 Options dialog box, select a USERCODE format (Hex, ASCII, or Binary) from the USERCODE Format drop-down list, then type alphanumeric values using up to 16 characters
   - For the default ASCII format, type alphanumeric values using up to 16 characters.
   - For hex format, type values of 0 through F, using up to 32 characters.
   - For binary format, type 0 and 1 values, using up to 128 characters.
8. Alternately, you can choose to overwrite USERCODE with JEDEC checksum by checking the box.
9. Click the **Encryption** box.
10. To edit the encryption key, click **Edit Key** to display the Encryption Key dialog box. Edit the encryption key as desired, or load an Encryption Key Format file (.bek or .BEK) from file, and click **OK**..

    Note

    If the encryption key was entered manually, you must save the key to a .BEK file.

11. Click **Next**.
12. In the Step 3 of 4 Select Output File(s) dialog box, select the desired output file type from the dropdown menu, and using the ... button, browse to the desired location of the output file(s).
13. Click **Next**.
14. In the Step 4 of 4 Generate Deployment dialog box, review Deployment Tool Summary and Command Line Information.
15. Choose **File > Generate**, or click the button, to perform the Deployment. The Deployment Generate Result are available for review.
16. If you wish to save the deployment, choose **File > Save**, or click the button, to save the deployment (.ddt) file in your desired location.

For more information about setting security options for Lattice FPGA devices, refer to the following Lattice Technical Notes:

- TN1212, LatticeXP2 Advanced Security Programming Usage Guide
- TN1215, Advanced Security Encryption Key Programming Guide for ECP5, LatticeECP3, and LatticeECP2/MS Devices

See Also ▶"Setting Security Options" on page 76

- "Encrypting a JEDEC with Deployment Tool" on page 1000

Authenticating a Bitstream with Deployment Tool (LFMNX and MachXO3D Only)

Authentication settings are initially set in the Diamond software using the Security Setting Tool. Refer to "Setting Security Options" on page 76.

Deployment Tool allows you to convert an existing normal bitstream (.bit or .rbt) to an authentication output bitstream.

IMPORTANT!

To turn on this security feature, you need to install the Encryption Security Control Pack.

To convert a normal .bit or .rbt file to an authenticated output bitstream with Deployment Tool:

1. Issue the start command.
   - In Windows choose **Programs > Lattice Diamond > Accessories > Deployment Tool**.
   - In Linux, from the <*install_path*>/bin/lin directory, enter the following on a command line:

     ```
     ./deployment
     ```
2. In the Getting Started dialog box, choose **Create a New Deployment.**
3. In the Function Type dropdown menu, choose **File Conversion**.
4. In the Output File Type box, choose **Bitstream**.
5. Click **OK**. The Deployment Tool loads the device database.
6. In the Step 1 of 4 dialog box, select Input File(s) window, click **... (Browse)** in the File Name box, and browse to the file you wish to authenticate, then click **Next**.
7. In the Step 2 of 4 Options dialog box, select a USERCODE format (Hex, ASCII, or Binary) from the USERCODE Format drop-down list, then type alphanumeric values using up to 16 characters.
   - For the default ASCII format, type alphanumeric values using up to 16 characters.
   - For hex format, type values of 0 through F, using up to 32 characters.
   - For binary format, type 0 and 1 values, using up to 128 characters.
8. Alternately, you can choose to overwrite USERCODE with JEDEC checksum by checking the box.
9. Click the **Authentication** box.
10. To load the public key, under Public Key, click **Load,** and in the security_setting directory, browse to desired public key (.pub) file.

11. Depending on which type of authentication method was selected in the Security Setting tool:
    - For Private Key, click **Load**, and in the security_setting directory, browse to desired Private Key (.prv) file. In the Encryption File Password dialog box, provide a password to open the file, and click **OK**.
    - For ECDSA Signature, click **Load**, and in the security_setting directory, browse to desired Private Key (.prv) file. In the Encryption File Password dialog box, provide a password to open the file, and click **OK**.
12. Click **Next**.
13. In the Step 3 of 4 Select Output File(s) dialog box, select the desired output file type from the dropdown menu, and using the ... button, browse to the desired location of the output file(s).
14. Click **Next**.
15. In the Step 4 of 4 Generate Deployment dialog box, review Deployment Tool Summary and Command Line Information.
16. Choose **File > Generate**, or click the button, to perform the Deployment. The Deployment Generate Result are available for review.
17. If you wish to save the deployment, choose **File > Save**, or click the button, to save the deployment (.ddt) file in your desired location.

See Also ▶"Setting Security Options" on page 76

Opening an Existing Deployment While Running the Deployment Tool

You can open a different existing deployment while running the Deployment Tool, and save the deployment you are currently working on.

To open an existing deployment while running the Deployment Tool:

1. Choose **File > Open**, or click the button, and browse to the deployment (.ddt) file that you wish to open.
2. Deployment Tool prompts you to save the .ddt file for the project you are currently working on. If you wish to save the deployment you are currently working on, click **Yes**, and in the Save As dialog box, create a name for the .ddt file and save in the desired location.
3. In the Open dialog box, choose the .ddt file you wish to open.
4. Click **Open**.

See Also ▶"Creating a New Deployment" on page 997
- "Input File Descriptions" on page 993
- "Output Options Descriptions" on page 997

- “Bitstream Output Formats” on page 996
- “Bitstream Output File Descriptions” on page 995

Creating a New Deployment While Running the Deployment Tool

You can create a new deployment while running the Deployment Tool.

To create a new deployment while running the Deployment Tool:

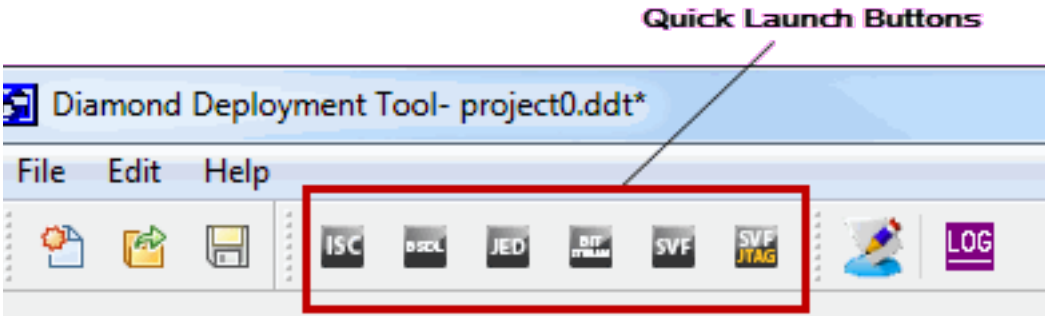
1. Choose **File > New**, or click the  button.
2. Deployment Tool prompts you to save the .ddt file for the project you are currently working on. If you wish to save the deployment you are currently working on, click **Yes**, and in the Save As dialog box, create a name for the .ddt file and save in the desired location.
3. In the Getting Started dialog box, choose **Create a New Deployment.**
4. In the Function Type dropdown menu, choose from one of the following options:
   - File Conversion
   - Tester
   - Embedded System
   - External Memory
5. In the Output File Type box, choose an output file type.
6. Click **OK**. The Deployment Tool loads the device database.

See Also “Creating a New Deployment” on page 997

- “Using Quick Launch Button to Create a New Deployment” on page 1005
- “Input File Descriptions” on page 993
- “Output Options Descriptions” on page 997
- “Bitstream Output Formats” on page 996
- “Bitstream Output File Descriptions” on page 995

Using Quick Launch Button to Create a New Deployment

Quick launch buttons allow you to create a new Deployment while running the Deployment Tool without having to use the Getting Started dialog box. The toolbar supports up to six quick launch buttons.



You can customize which quick launch buttons appear in the toolbar. Choices of quick launch buttons include:

- IEEE 1532 ISC Data File
- Application Specific BSDL File
- JEDEC File
- Bitstream
- SVF - Single Device
- SVF - JTAG Chain
- STAPL - Single Device
- STAPL - JTAG Chain
- ATE
- JTAG Full VME Embedded
- JTAG Slim VME Embedded
- Slave SPI Embedded
- I2C Embedded
- sysCONFIG Embedded
- Hex Conversion
- Dual Boot
- Advanced SPI Flash
- sysCONFIG Daisy Chain

To customize which quick launch buttons appear in the Deployment Tool toolbar:

1. Place cursor in the toolbar, and right-click.
2. In the drop-down menu, check the box of the quick start button that you want to display in the toolbar. A maximum of six buttons may be selected.

See Also

- “Creating a New Deployment” on page 997

- "Creating a New Deployment While Running the Deployment Tool" on page 1005
- "Input File Descriptions" on page 993
- "Output Options Descriptions" on page 997
- "Bitstream Output Formats" on page 996
- "Bitstream Output File Descriptions" on page 995

Viewing the Deployment Tool Log File

The log file shows recent actions in a text editor.

To view the log file:

Click LOG button on the toolbar.

See Also "Deploying the Design with the Deployment Tool" on page 950

Changing the Deployment Tool Log File Settings

The log file shows recent actions in a text editor. You can change settings to clear log file path each time the application starts, and you can also change where the log file is saved.

To change where the log file is saved:

1. Choose **Edit > Settings**.
2. In the Settings dialog box, browse to the folder where you wish to save the deployment_tool.log file.

To clear log file each time the application starts:

1. Choose **Edit > Settings**.
2. In the Settings dialog box, check the **Clear Log File Each Time Application Starts** box.

If the **Clear Log File Each Time Application Starts** box is unchecked, the log file will continue increase in size until the file is manually erased.

See Also "Deploying the Design with the Deployment Tool" on page 950

Debugging SVF, STAPL, and VME Files

Download Debugger is a stand-alone software tool for debugging Serial Vector Format (SVF) files, Standard Test And Programming Language (STAPL) files, and Lattice Embedded (VME) files. Download Debugger allows you to program a device, and edit, debug, and trace the process of SVF, STAPL, and VME files.

Download Debugger also allows you to create, edit, or view a VME file in hexadecimal format.

This section provides procedures for using Download Debugger. Topics include:

- "Understanding SVF Files" on page 1009
- "Download Debugger Software Support of SVF Operations" on page 1009
- "Running Download Debugger" on page 1010
- "Opening an Existing SVF File" on page 1010
- "Creating a New SVF File" on page 1011
- "Setting Device Programming Options in Download Debugger" on page 1011
- "Setting Port Assignments and Options in Download Debugger" on page 1011
- "Viewing SVF Processing" on page 1011
- "Setting and Removing Breakpoints in an SVF File" on page 1012
- "Processing an SVF File" on page 1013
- "Viewing the Download Debugger Log File" on page 1013
- "Saving a Log File in Download Debugger" on page 1014
- "Clearing the Contents of a Log File in Download Debugger" on page 1014
- "Editing an SVF File" on page 1014
- "Setting Windows Options in Download Debugger" on page 1014
- "Opening an Existing STAPL File" on page 1015
- "Creating a New STAPL File" on page 1015
- "Viewing STAPL Processing" on page 1015
- "Setting and Removing Breakpoints in a STAPL File" on page 1016
- "Processing a STAPL File in Download Debugger" on page 1017
- "Viewing the Download Debugger Log File" on page 1017
- "Saving a Log File in Download Debugger" on page 1018
- "Clearing the Contents of a Log File in Download Debugger" on page 1018
- "Editing a STAPL File" on page 1018

▶ "Setting Windows Options in Download Debugger" on page 1019

Understanding SVF Files

The Serial Vector Format file (.svf) is the medium for exchanging descriptions of high-level 1149.1 bus operations. The SVF file is defined as an ASCII file that consists of a set of SVF statements. The 1149.1 bus operations consist of scan operations and movements between deferent stable states. Refer to the Serial Vector Format Specification Rev E for detailed definitions of SVF statement formats. Current SVF specifications are available from Asset-Intertech website at www.asset-intertech.com.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Download Debugger Software Support of SVF Operations

The Download Debugger supports a selective set of SVF operations. The following table lists SVF operations supported by the Download Debugger.

SVF Operation	Description	Support Status
ENDDR	Specifies default end state for DR scan operations.	Full support
ENDIR	Specifies default end state for IR scan operations.	Full support
HDR	(Header Data Register) Specifies a header pattern, which is placed at the beginning of subsequent DR, scan operations.	Supports TDI keyword only. SMASK, TDO and MASK are not supported.
HIR	(Header Instruction Register) Specifies a header pattern, which is placed at the beginning of subsequent IR, scan operations.	Supports TDI keyword only. SMASK, TDO and MASK are not supported.
RUNTEST	Forces the 1149.1 bus to the RUN_TEST/IDLE state for a specified number of clocks.	Full support
SDR	(Scan data register) Performs an 1149.1 data register scan.	Supports TDI, TDO, and MASK keywords. SMASK is ignored.
SIR	(Scan Instruction Register) Performs an 1149.1 instruction register scan.	Supports TDI, TDO, and MASK keywords. SMASK is ignored.
STATE	Forces the 1149.1 bus to a specified stable state.	Supports only single target state. The Download Debugger does not allow you to specify custom traverse path.

SVF Operation	Description	Support Status
TDR	(Trailer Data Register) Specifies a trailer pattern, which is appended to the end of subsequent DR scan operations.	Supports TDI keyword only. SMASK, TDO, and MASK are not supported.
TIR	(Trailer Data Register) Specifies a trailer pattern which is appended to the end of subsequent IR scan operations	Supports TDI keyword only. SMASK, TDO, and MASK are not supported.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Running Download Debugger

Download Debugger runs separately from the Diamond software environment.

To start the Download Debugger:

1. Issue the start command.
   - In Windows choose **Programs > Lattice Diamond > Accessories > Download Debugger**.
   - In Linux, from the <*install_path*>/bin/lin directory, enter the following on a command line:

     ```
     ./debugger
     ```

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Opening an Existing SVF File

The Download Debugger Edit Window allows you to perform various editing functions on your SVF file. You can check the SVF file for errors line by line when you program your device.

To open an existing SVF file:

1. In Download Debugger, choose **File > Open** to open the dialog box.
2. Locate and select the desired SVF file, and then click **Open**.

   The selected SVF file opens in the Download Debugger Edit Window.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Creating a New SVF File

To create a new empty SVF file:

- In Download Debugger, choose **File > New** to open the SVF Debugger Edit Window and create a new empty SVF file. The Options Dialog Box also opens.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Setting Device Programming Options in Download Debugger

To set SVF options:

1. In Download Debugger, choose **Configuration > Options**.

   The "Options Dialog Box" on page 1020 opens.

2. Select the options that you want and click **OK**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Setting Port Assignments and Options in Download Debugger

Use the "Cable and I/O Port Settings Dialog Box" on page 1019 to specify the preferred parallel port for download, select the port address, and select the download cable type.

To set port assignments and options:

1. In Download Debugger, choose **Configuration > Cable and I/O Port Setup**.

   The "Cable and I/O Port Settings Dialog Box" on page 1019 opens.

2. Select the options that you want and click **OK**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Viewing SVF Processing

To view the progress of SVF processing, you need to open the Output dialog box before processing the SVF file.

To view the progress of SVF processing:

- In Download Debugger, choose **Configuration > SVF Output** before processing the SVF file.

  The Output Dialog Box opens, displaying the programming pin values, clock pulses, data register, and JTAG state. Keep the dialog box open. The display will be refreshed in real time during SVF processing.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Setting and Removing Breakpoints in an SVF File

By using the Download Debugger, you can set a breakpoint in your SVF file before processing it. The process stops at the line that you specify, and a dialog box notifies you that the breakpoint has been reached. After clicking OK in the dialog box, you can resume the process.

To set a breakpoint in an SVF File:

1. Select a line in your SVF file.
2. Choose **Command > Breakpoint > Set Breakpoint**.

To select the next breakpoint:

1. Select a line in your SVF file.
2. Choose **Command > Breakpoint > Next Breakpoint**.

To select the previous breakpoint:

1. Select a line in your SVF file.
2. Choose **Command > Breakpoint > Previous Breakpoint**.

To clear all breakpoints:

1. In your SVF file, select the breakpoint you want to remove.
2. Choose **Command > Breakpoint > Clear All Breakpoints**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Processing an SVF File

After setting device programming options and port assignments, you can process your SVF file with the Download Debugger and examine it in the SVF Output box. You can program a device with a download cable, step through each line of the process, reset it and begin again.

Note

To view the SVF Output box, you must choose Configuration > SVF Output before processing the SVF file.

To process an SVF file:

- Choose **Command > Go**.

To step through a process, line by line:

- Choose **Command > Step**.

  The process will step through the SVF file line by line.

To step the state machine and toggle the programming signal:

1. Choose **Command > Mini Step**.
2. In the Manual TAP Operations Dialog Box, select state path as you wish, and step through the TAP controller.

To reset a process and start a process from the beginning:

- Choose **Command > Reset**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Viewing the Download Debugger Log File

The Download Debugger generates a log file of the process and reports any errors encountered. When you select **Continue on Error** in the "Options Dialog Box" on page 1020, the log file also reports state machine transitions and delay time for debug.

To view a log file:

- In Download Debugger, choose **View > View Log File**.

  The log file is displayed in a text editor.

See Also "Clearing the Contents of a Log File in Download Debugger" on page 1014

- "Debugging SVF, STAPL, and VME Files" on page 1008

Saving a Log File in Download Debugger

To save a log file:

1. In Download Debugger, choose **Configuration > Options**.
2. The "Options Dialog Box" on page 1020 opens.
3. Specify the path to the directory that you wish to save your log file. You can use the Browse button to navigate to the directory where you would like to save the log file.
4. Click **OK**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Clearing the Contents of a Log File in Download Debugger

To clear the contents of the log file:

- In Download Debugger, choose **Edit > Clear Log File**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Editing an SVF File

Use the Edit menu commands in the Download Debugger to edit an SVF file.

To edit an SVF file:

1. In Download Debugger, select a line in your SVF file.
2. Make the edits using commands in the Edit menu, and click **File > Save**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Setting Windows Options in Download Debugger

Use the Windows options to change the way the Download Debugger software is displayed.

To cascade open windows:

- Choose **Window > Cascade**.

To tile open windows:

- Choose **Window > Tile**.

To arrange minimized open windows:

- Choose **Window > Arrange Icons**.

  Minimized open windows in Download Debugger will be arranged along the bottom of your computer screen.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Opening an Existing STAPL File

The Download Debugger allows you to perform various editing functions on your STAPL file. You can check the STAPL file for errors line by line when you program your device.

To open an existing STAPL file:

1. In Download Debugger, choose **File > Open** to open the dialog box.
2. Locate and select the desired STAPL (.stp or .jam) file, and then click **Open**.

   The selected STAPL file opens in the Download Debugger.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Creating a New STAPL File

To create a new empty STAPL file:

- In Download Debugger, choose **File > New**.

  A blank edit window opens. You can start entering contents for your new STAPL file, and save the file when you finish.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Viewing STAPL Processing

To view the output information during STAPL processing, you should open the Output Information dialog box before processing the STAPL file.

To view the progress of STAPL processing:

- In Download Debugger, choose **Configuration > STAPL Output** before processing the STAPL file.

The Output Information Dialog Box opens, displaying the programming pin values, clock pulses, data register, JTAG state, and variables values. Keep the dialog box open. The display will be refreshed in real time during STAPL processing.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Setting and Removing Breakpoints in a STAPL File

By using the Download Debugger, you can set a breakpoint in your STAPL file before processing it. The process stops at the line that you specify, and a dialog box notifies you that the breakpoint has been reached. After clicking OK in the dialog box, you can resume the process.

To set a breakpoint in a STAPL file:

1. Select a line in your STAPL file.
2. Choose **Command > Breakpoint > Set Breakpoint**.

To select the next breakpoint:

1. Select a line in your STAPL file.
2. Choose **Command > Breakpoint > Next Breakpoint**.

To select the previous breakpoint:

1. Select a line in your STAPL file.
2. Choose **Command > Breakpoint > Previous Breakpoint**.

To clear all breakpoints:

1. In your STAPL file, select the breakpoint you want to remove.
2. Choose **Command > Breakpoint > Clear All Breakpoints**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Processing a STAPL File in Download Debugger

After setting device programming options and port assignments, you can process your STAPL file with the Download Debugger and examine the progress in the Output Information dialog box.You can program a device with a download cable, step through each line of the process, reset it and begin again.

Note

To view the Output Information dialog box, you must choose **Configuration > STAPL Output** before processing the STAPL file.

To process a STAPL file:

- Choose **Command > Go**.

To step through a process, line by line:

- Choose **Command > Step**.

  The process will step through the STAPL file line by line.

To step the state machine and toggle the programming signal:

1. Choose **Command > Mini Step**.
2. In the Manual TAP Operations Dialog Box, select state path as you like, and step through the TAP controller.

To reset a process and start a process from the beginning:

- Choose **Command > Reset**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Viewing the Download Debugger Log File

The Download Debugger generates a log file that tracks the process and reports any errors encountered. The log file contains all the information displayed in the Status window and also the status of the cable. When you select **Continue on Verification Failure** in the "Options Dialog Box" on page 1020, the log file also reports state machine transitions and delay time for debug.

To view a log file:

- In Download Debugger, choose **View > View Log File**.

  The log file is displayed in a text editor.

See Also "Saving a Log File in Download Debugger" on page 1014

- "Clearing the Contents of a Log File in Download Debugger" on page 1018
- "Debugging SVF, STAPL, and VME Files" on page 1008

Saving a Log File in Download Debugger

To save a log file:

1. In Download Debugger, choose **Configuration > Options**.

   The "Options Dialog Box" on page 1020 opens.
2. Specify the path to the directory that you want to save your log file. You can use the Browse button to navigate to that directory.
3. Click **OK**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Clearing the Contents of a Log File in Download Debugger

To clear the contents of the log file:

- In Download Debugger, choose **Edit > Clear Log File**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Editing a STAPL File

Use the Edit menu commands in the Download Debugger to edit a STAPL file.

To edit a STAPL file:

1. In Download Debugger, select a line in your STAPL file.
2. Make the edits using commands in the Edit menu, and click **File > Save**.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Setting Windows Options in Download Debugger

Use the Windows options to change the way how multiple windows are displayed in Download Debugger.

To cascade open windows:

- Choose **Window > Cascade**.

To tile open windows:

- Choose **Window > Tile**.

To arrange minimized open windows:

- Choose **Window > Arrange Icons**.

  Minimized open windows in Download Debugger will be arranged along the bottom of your computer screen.

See Also "Debugging SVF, STAPL, and VME Files" on page 1008

Download Debugger Options

This section lists the options available in Download Debugger.

Topics include:

- "Cable and I/O Port Settings Dialog Box" on page 1019
- "Options Dialog Box" on page 1020

Cable and I/O Port Settings Dialog Box

The following options are available in the Cable and I/O Port Settings dialog box.

Cable Description This option is displayed when multiple cables are detected. When enabled, this option allows you to select the desired cable.

Cable Specifies the download cable type: Lattice (parallel port), USB (LSC USB cable), or USB2 (FTDI USB2 cable)

Port Specifies the port to which the download cable is connected.

Custom Port Specifies a custom parallel port to which the download cable is connected. Use hexadecimal format to type the port name.

Using Slave SPI Interface Connection This option allows you to target the slave SPI port of the selected device. This allows you to debug slave SPI device programming

Use Default I/O Settings Only use the four JTAG signals.

Use Custom I/O Settings .Specify additional non-JTAG signals connected to the board.

INITN Pin Connected Select this option if you connect the INIT pin to the download cable. This option is only available with the ispDOWNLOAD USB cable.

DONE Pin Connected Select this option if you connect the DONE pin to the download cable. This option is only available with the ispDOWNLOAD USB cable.

TRST Pin Connected Select this option if you connect the TRST pin to the download cable. Specify active high or active low.

PROGRAM Pin Connected Select this option if you connect the PROGRAMN pin to the download cable.

ispEN Pin Connected Select this option if you connect the ispEN pin to the download cable. Specify active high or active low.

See Also "Download Debugger Options" on page 1019

Options Dialog Box

The following options are available in the Options dialog box.

Log Allows you to specify name and location of log file.

Clear Log File Each Time Start Application A Check to clear the log file each time Download Debugger is started.

Source Editor Allows you to specify font, size, tab size, of Source Editor.

Show The Number Toggles display of line numbers in left margin of Source Editor.

Show Indicator Margin Toggles display of indicator margin in left side of Source Editor.

Disable SVF Syntax Checker Disables the tool that checks the syntax of the SVF file.

Ignore IR and DR Header Trailer in SVF Ignores the HIR, TIR, HDR, and TDR information in SVF file. If you set up HIR, TIR, HDR, or TDR in this dialog box, this option must be selected.

Continue on Error Continues processing the file even if there is an error. You can look at the log file for the errors encountered. When this option is selected, the log file also reports state machine transitions and delay time for debug.

Mixed Chain Tells the Download Debugger that the hardware configuration is a mixed chain, therefore disabling the ISP chain when processing the SVF file.

SVF TCK Frequency Specifies the TCK clock frequency for the chosen SVF file. This value is used to determine the TCK clock period and the length of the delay times.

SVF Vendor Allows you select from the following SVF vendors for the selected device: JTAG STANDARD, LATTICE, ALTERA, and XILINX. This option is available for JTAG devices only.

Instruction Register Header Sets the IR header length in number of bits (0-100).

Instruction Register Trailer Sets the IR trailer length in number of bits (0-100).

Data Register Header Sets the HDR in number of bits (0-100).

Data Register Trailer Sets the TDR in number of bits (0-100).

Ignore IR and DR Header Trailer in STAPL Ignores the HIR, TIR, HDR and TDR information in STAPL file. If you set up HIR, TIR, HDR, or TDR in this dialog box, this option must be selected.

Starting TAP State Select the starting JTAG state of the JTAG State Machine for download. Only two states are available: TLR (Test-Logic-Reset) and RTI (Run-Test/Idle).

See Also “Download Debugger Options” on page 1019

Using the Model 300 Programmer

The Model 300 Programmer is a simple engineering device programmer that allows you to perform single-device programming directly from a PC or Linux environment. The Model 300 Programmer software and hardware support all JTAG devices produced by Lattice, with device Vcc of 1.8, 2.5, 3.3, and 5.0V.

Programming Software Support

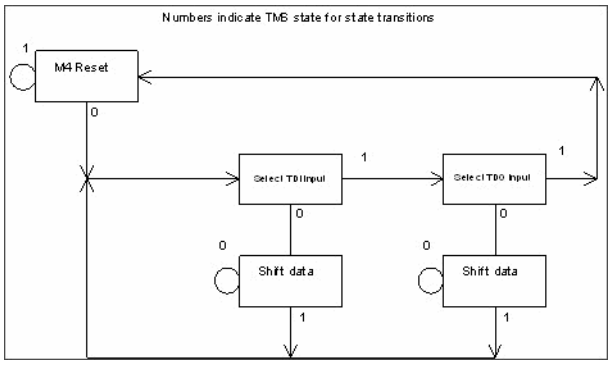
The Model 300 Programmer software controls the programming process. When you select a device, the programming adapter configuration and the correct device Vcc is automatically set up. LEDs indicate the power status and the selected Vcc level.

Software Control

The Model 300 programmer is controlled by the TRST, ENABLE (ispEN), TMS, TDI, and TCK signals from the download cable. When in normal operation mode, the Model 300 programmer controls IC channels TMS, TDI, TCK, and TDO to/from the programming adapter socket. TRST and ENABLE control the programmer's operation mode.

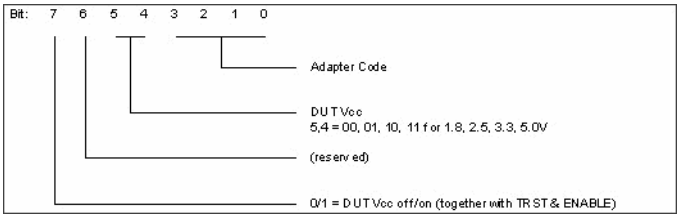
TRST	ENABLE	Description
0	0	DUT is target of JTAG signals, DUT Vcc is off
0	1	DUT is target of JTAG signals, DUT Vcc is on if Control Bit 7 = 1
1	0	Control IC TAP is target of JTAG signals, DUT Vcc is off
1	1	Control IC State Machine is target of JTAG signals, DUT Vcc is off when State Machine is not in M4 Reset state

When TRST and ENABLE are high and LOCAL is Off, JTAG signals interact with the State Machine programmed into the M4A5-64/32 Control IC. Use this State Machine to write or read Control Register bits for routing the JTAG signals to the appropriate pins for the selected adapter and for controlling DUT Vcc.



When TDI input is selected, data is shifted into the State Machine from TDI, to write new data into the Control Register. When TDO input is selected, data is rotated through the Control Register to read it out on TDO. It must be shifted eight times to return the Control Register to its original pattern.

Control Register



Valid Adapter Code values are 0000, 0001, 0010, 0011, 0100, and 1100.

Device Support

All Lattice products that feature non-volatile configuration elements are supported. This includes devices with a VCC of 1.2V, 1.8V, 2.5V, 3.3V and 5V.

Socket Support

Individual devices are supported via device/package-specific socket adapters, which interface to the 28-pin DIP on the Model 300. For a complete list of supported devices and socket adapters, see the Socket Adapter list on the Lattice web site.

Connector Support

The interface connection from the controlling PC is through a 10-pin JTAG connector on the Model 300. This is the standard configuration for Lattice's ispDOWNLOAD 10-pin JTAG connector cable.

Power Supply Support

The Model 300 programmer requires 9V DC at 1A minimum, to provide power for the programmer itself and for programming the target device. A 9V power adapter is included with each kit. This power adapter accepts 110V to 220V inputs, and includes interchangeable physical plugs for use worldwide.

See Also ▶"Setting Up the Hardware" on page 1024

- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029

- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Setting Up the Hardware

Before you can use the Model 300 programmer, you must connect a download cable to your PC and insert the socket adapter.

To set up your Model 300 programmer:

1. Turn on and boot up the PC, if you have not already done so.
2. With the Model 300 programmer in the power off position, connect the download cable between the PC parallel port and the Model 300 programmer.
3. Connect the power supply to a properly grounded AC outlet and to the AC plug at the rear of the Model 300 programmer.
4. Make sure that the PC has booted up completely, and then turn on the Model 300 programmer.
5. Place the device in the programming adapter socket. The 1.8V and PWRRDY LED indicators illuminate, indicating that the Model 300 is ready for programming.

See Also "Using the Model 300 Programmer" on page 1022

- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Running the Model 300 Software

To run the Model 300 programmer:

1. Make sure that the Model 300 programmer is turned on and connected to the PC by the appropriate download cable.

2. Issue the start command.

   - In Windows choose **Programs > Lattice Diamond > Accessories > Model 300 Programmer**.
   - In Linux, from the <*install_path*>/bin/lin directory, enter the following on a command line:

     ```
     ./model300
     ```

   The Model 300 Programmer Window opens.

See Also "Using the Model 300 Programmer" on page 1022

- "Setting Up the Hardware" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Programming a Device

To program a device:

1. If you are using a 28-pin socket adapter, insert the socket adapter into the 40-pin ZIF socket of the Model 300. Make sure that the socket adapter is bottom-justified.

   Note

   If the socket adapter is not bottom-justified, Model 300 will not read or program the device.

   If you are using a Lattice socket adapter board, insert the socket adapter board pins into the appropriate connector on both sides of the 40-pin ZIF socket of the Model 300. Make sure that all of the pins are installed correctly.

2. Using proper ESD handling procedures, place the device in the programming adapter socket and ensure the correct alignment of the device.
3. In the Model 300 main window, select the appropriate device, device family, package, Model 300 adapter, and Operation.
4. Under File Name, browse select the required data file.

5. Choose **Design > Program, or** click  on the toolbar, to download the program to the device.

   The Vcc ON indicator on the Model 300 Programmer turns red during the download process. It turns off when the operation is completed. The Output Pane of the Model 300 software indicates the progress of the operation and reports any errors. A green "Pass" message in the Status field shows whether the operation was successful.

6. When the process has finished, remove the device from the adapter socket and, optionally, turn off the Model 300 Programmer.

See Also ▶"Using the Model 300 Programmer" on page 1022

- "Setting Up the Hardware" on page 1024
- "Running the Model 300 Software" on page 1024
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Programming a JTAG-SVF or JTAG-NOP Device

You can program any JTAG-SVF or JTAG-NOP with the Model 300 Programmer and software.

To program a JTAG-SVF or JTAG NOP device:

1. Make sure that the 28-pin socket adapter has been correctly inserted into the Model 300 Programmer and that the device has been placed into the adapter socket with the correct alignment.
2. In the Model 300 main window, under Device Family, select **Generic JTAG Device**.
3. Under Device, select **JTAG-SVF** or **JTAG-NOP**.
4. Select the appropriate device package, Model 300 adapter, and Operation.
5. Under File Name, browse select the required SVF file.
6. Choose **Design > Program**, or click  , to download the program to the device.

The Vcc ON indicator on the Model 300 Programmer turns red during the download process. It turns off when the operation is completed. The Output Pane of the Model 300 software indicates the progress of the operation and reports any errors. A green "Pass" message in the Status field shows whether the operation was successful.

When the process has finished, remove the device from the adapter socket and, optionally, turn off the Model 300 Programmer.

See Also ▶"Using the Model 300 Programmer" on page 1022

- "Setting Up the Hardware" on page 1024
- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Saving a Model 300 Project

When you set up a Model 300 project for programming, you can save the Model 300 project as a .job file.

To save a Model 300 project:

1. Choose **File > Save Untitled**, or click  , and in the Save Model 300 File As dialog box, navigate to the desired directory.
2. Type a name in the file name box and click Save. The information entered in your project is saved in the .job file.

See Also ▶"Using the Model 300 Programmer" on page 1022

- "Setting Up the Hardware" on page 1024
- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030

▶ “Model 300 Options” on page 1031

Opening an Existing Model 300 Project

To open an existing Model 300 project:

1. Choose **File > Open file**, or click  .
2. In the Open Model 300 Programmer Project File dialog box, navigate to the desired Model 300 Project File (.job) and click **Open**.

See Also ▶“Using the Model 300 Programmer” on page 1022

▶ “Setting Up the Hardware” on page 1024

▶ “Running the Model 300 Software” on page 1024

▶ “Programming a Device” on page 1025

▶ “Programming a JTAG-SVF or JTAG-NOP Device” on page 1026

▶ “Saving a Model 300 Project” on page 1027

▶ “Editing Device Properties” on page 1028

▶ “Viewing the Log File” on page 1029

▶ “Selecting Project Settings” on page 1029

▶ “Checking the Model 300 Project” on page 1030

▶ “Model 300 Options” on page 1031

Editing Device Properties

To edit device properties:

1. In the Model 300 software main window, select the device that you want to edit.
2. Choose **Edit > Device Properties**, click  on the toolbar, or right-click and choose **Device Properties**.
3. In the Device Properties Dialog Box, edit the device properties.
4. Click **OK** to close the Device Properties Dialog Box.

See Also ▶“Using the Model 300 Programmer” on page 1022

▶ “Setting Up the Hardware” on page 1024

▶ “Running the Model 300 Software” on page 1024

▶ “Programming a Device” on page 1025

▶ “Programming a JTAG-SVF or JTAG-NOP Device” on page 1026

▶ “Saving a Model 300 Project” on page 1027

- “Opening an Existing Model 300 Project” on page 1028
- “Viewing the Log File” on page 1029
- “Selecting Project Settings” on page 1029
- “Checking the Model 300 Project” on page 1030
- “Model 300 Options” on page 1031

Viewing the Log File

The log file shows recent actions in a text editor. This allows you to check such things as the last time you performed a programming operation, the device you chose, and the results of the operation.

To view the log file:

- Choose Design > Log, or click  on the toolbar

See Also “Using the Model 300 Programmer” on page 1022

- “Setting Up the Hardware” on page 1024
- “Running the Model 300 Software” on page 1024
- “Programming a Device” on page 1025
- “Programming a JTAG-SVF or JTAG-NOP Device” on page 1026
- “Saving a Model 300 Project” on page 1027
- “Opening an Existing Model 300 Project” on page 1028
- “Editing Device Properties” on page 1028
- “Selecting Project Settings” on page 1029
- “Checking the Model 300 Project” on page 1030
- “Model 300 Options” on page 1031

Selecting Project Settings

Use the Project Settings command to specify how the software will download the program to the target device.

To select project settings:

1. In Model 300 Programmer, choose **Edit > Settings**.

   The “Settings Dialog Box” on page 1031 opens.
2. Select the appropriate check boxes for the specifications you want to include.
3. Choose between TLR and RTI for the starting and stopping TAP states.

4. To change the TCK clock pulse for an optional JTAG signal, click **Use Custom Pulse Width Delay**, and then type a different level in the TCK Low Pulse Width Delay box.
5. Click **OK**.

See Also ▶"Using the Model 300 Programmer" on page 1022

- "Setting Up the Hardware" on page 1024
- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Checking the Model 300 Project

To check the Model 300 project:

In Model 300 Programmer, choose **Design > Check Model300 Project** or click  on the toolbar

A message displays in the output pane stating whether or not the project is valid.

See Also ▶"Using the Model 300 Programmer" on page 1022

- "Setting Up the Hardware" on page 1024
- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Model 300 Options" on page 1031

Model 300 Options

This section lists the options available in Model 300. Topics include:

- "Device Properties Dialog Box" on page 1031
- "Settings Dialog Box" on page 1031

Device Properties Dialog Box

The following options are available in the Device Properties dialog box:

Access Mode Selects the mode for programming the device.

Operation Lists the available operation modes for the device. Select one from the drop-down list.

Programming File Selects the data file for programming or verify.

Reinitialize Part on Program Error Reinitializes the part to its blank, or erased, state when a programming error occurs.

See Also

- "Using the Model 300 Programmer" on page 1022
- "Setting Up the Hardware" on page 1024
- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Settings Dialog Box

The following options are available in the Settings dialog box:

USERCODE Display Allows the user to choose in which format the USERCODE will be displayed. Choices are Hex, ASCII, and Decimal.

Log File Path Shows the location of the Programmer log file.

Use Default JTAG States (TLR/TLR) Uses Test-Logic-Reset (TLR) as the starting and ending JTAG state of the JTAG State Machine for download.

Avoid Test Logi Reset (TLR) State Avoids the Test Logic Reset State of the JTAG State Machine during download.

Use Custom JTAG States Specify Test-Logic-Reset (TLR) or Run-Test/Idle (RTI) as the starting and ending JTAG state of the JTAG State Machine for download.

Initial TAP State Specify TLR or RTI as the starting JTAG state of the JTAG State Machine for download.

Final TAP State Specify TLR or RTI as the ending JTAG state of the JTAG State Machine for download.

Use Default Pulse Width Delay Uses the fastest TCK clock speed.

Use Custom Pulse Width Delay Enables the Use Custome Pulse Width Delay feature.

TCK Low Pulse Width Delay (1 - 10x) Allows you to slow down the TCK clock. This is done by extending the low period of the clock.

Check Cable Setup Before Programming Confirms that the cable signals are correctly connected to the board and devices in the JTAG chain on the board match the devices selected in the .xcf file.

Continue Download on Error Ignores any errors while downloading and continues running.

See Also

- "Using the Model 300 Programmer" on page 1022
- "Setting Up the Hardware" on page 1024
- "Running the Model 300 Software" on page 1024
- "Programming a Device" on page 1025
- "Programming a JTAG-SVF or JTAG-NOP Device" on page 1026
- "Saving a Model 300 Project" on page 1027
- "Opening an Existing Model 300 Project" on page 1028
- "Editing Device Properties" on page 1028
- "Viewing the Log File" on page 1029
- "Selecting Project Settings" on page 1029
- "Checking the Model 300 Project" on page 1030
- "Model 300 Options" on page 1031

Using Programming File Utility

The Programming File Utility is a stand-alone tool that allows you to view, compare, and edit data files. When comparing two data files, the software generates an output (.out) file with the differences highlighted in red.

See Also ▶"Running Programming File Utility" on page 1033

- "Viewing Data Files" on page 1033
- "Comparing Two Data Files" on page 1034
- "Editing Feature Row Values" on page 1035
- "Editing Control Register0 Values" on page 1035
- "Editing the USERCODE in the Data File" on page 1036
- "Securing the Device and Setting the Persistent Bit" on page 1037

Running Programming File Utility

To Run the Programming File Utility:

1. Issue the start command.
   - In Windows choose **Programs > Lattice Diamond > Accessories > Programming File Utility**.
   - In Linux, from the <*install_path*>/bin/lin directory, enter the following on a command line:

     ```
     ./fileutility
     ```

     The Programming File Utility window opens.

See Also ▶"Using Programming File Utility" on page 1033

Viewing Data Files

To view a data file:

- In the Programming File Utility, choose **File > Open** and select the data file you want to view.

  Note

  The data file can be in any of the following formats: .jed, .isc, .rbt, .bit, .rbk, .rbka, .msk, .mska.

Use the **Edit > Find** and **Edit > Find Again** commands to search for text in the file. Choose **File > Save As** to save the file with another file name.

Change file appearance such as font and font size by choosing **Configuration > Options** and changing settings in the Options dialog box.

See Also ▶"Using Programming File Utility" on page 1033

Comparing Two Data Files

You can use the Programming File Utility application to compare two data files. After creating the output file, the software displays the output file and highlights the differences between the two files.

To compare two data files:

1. Choose **Command > Data File Comparison** or click  on the toolbar.

   The Data Files Comparison Dialog Box opens.
2. Click the Browse buttons under First Data and Second Data to select the two data files you want to compare.
3. Optionally, if you are comparing two readback files (.rbk), browse for the mask file (.msc) to include masked bits in the comparison.
4. If desired, type a different name for the output file or browse to a file name that you want to overwrite. By default, the software uses the first data file name and appends the extension .out.
5. Click **OK**.

   The software compares the files and displays a message indicating whether the two files are identical or contain differences.
6. Click **OK** to close the message box and display the output file in the Programming File Utility window.
7. If the files contain differences, the Next  and Previous  arrows will become enabled on the toolbar. Use these buttons to examine each difference in the output file.

Viewing the Output File

The output file displays lines of data from the first file. Each of these lines is followed by a comparison line. The comparison line shows dots where the files are the same; where different, the line is highlighted in red and the data from the second file is displayed.

An output file that uses a mask file to compare two readback files displays each line of data from the first readback file in black, followed by a line of data in green from the mask file, followed by a comparison line. The comparison line shows dots where the two readback files are the same; where different, the line is highlighted in red and the data from the second readback file is displayed.

See Also ▶"Using Programming File Utility" on page 1033

Editing Feature Row Values

You can use the Feature Row Editor to enable or disable silicon features in the MachXO2, MachXO3L, and Platform Manager 2 devices by editing the Feature Row fuse settings in the data file.

To edit control register values:

1. Choose **Tools > Feature Row Editor**.

   The "Feature Row Editor Dialgog Box" on page 1038 opens.

2. Using the **(Browse)** button, browse to the data (.jed) file you want to edit.

3. Click **Read** to display current data file settings.

   The software displays the default values in the top row. On the second row, it displays in block the values that can be modified, and in red the values that cannot be modified.

4. Click the cell of a value displayed in black to change it.

   - To change the value in Feature Row, click the value cell and toggle the value from 1 to 0 or from 0 to 1.

5. To overwrite the existing data file, choose **Save**.

6. To create a new data file with a different name, choose **Save As** and save as a different file name.

See Also "Using Programming File Utility" on page 1033

Editing Control Register0 Values

You can use Control Register Editor0 to read the control registers of a bitstream file, modify the settings, and save them.

To edit control register values:

1. Choose **Tools > Control Register0 Editor**.

   The "Control Register0 Editor Dialog Box" on page 1038 opens.

2. Using the **(Browse)** button, browse to the .jed, .isc, .rbt, or .bit file that you want to edit, select the file, and then click **Open**.

   The editor opens with the selected file.

3. Click **Read** to display the values of the control register.

   The software displays the default values in the top row. On the second row, it displays in block the values that can be modified, and in red the values that cannot be modified.

4. Click the cell of a value displayed in black to change it.

- To change a value, click the value cell and toggle the value from 1 to 0 or from 0 to 1.
- To change the frequency in Control Register 0, click the Frequency value cell and select a frequency from the drop-down list.

5. To overwrite the existing data file, choose **Save.**
6. To create a new data file with a different name, choose **Save As** and save as a different file name.
7. A message box pops up, displaying the new control register value. Click **OK**.
8. Another message box tells you that Write to file was successful. Click **OK**.
9. Click **Close** to exit the Control Register.

See Also ▶"Using Programming File Utility" on page 1033

Editing the USERCODE in the Data File

The USERCODE Editor allows you to add information, such as serial code number, design code, or lot number, to the .jed, .isc, .rbt, or .bit file. This information is written to the USERCODE after the fuse map data in a JEDEC file. The signature must be in either decimal (0-9), ASCII, or hexadecimal (0-F) characters. In an ISC data file, this information is written in the user code data record in hexadecimal.

To use the USERCODE Editor:

1. Choose **Tools > USERCODE Editor**.

   The "USERCODE Editor Dialog Box" on page 1038 opens.

2. Using the **(Browse)** button, browse to the .jed, .isc, .rbt, or .bit file that you want to edit, select the file, and then click **Open**.

   The editor opens with the selected file.

3. Click **Read**.

   If there is no data in the field, you will see a string of 8 Fs (Hexadecimal).

4. Edit the string as required. The USERCODE edit box displays the current bit position as you type. The signature must be stated in decimal (0-9), ASCII, or hexadecimal (0-F) characters.
5. When you have finished, click **Save** or **Save As** to save the USERCODE data.
6. A message box asks you if you want to overwrite the data file? Click **Yes**.
7. Another message box tells you that the USERCODE write to file was successful. Click **OK**.

If you open the .jed file in a text editor, you will see the new USERCODE data added to the bottom of the file, as shown in the example below:

```
N User Electronic Signature
Data*U00000000000000000000000010011110011001100*
```

If you open the .isc data file in a text editor, you will see the new USERCODE data added to the end of the bottom of the file, as shown in the example below:

```
data usercode (initialize --LATTICE_NOTEDATA_WIDTH = 32--
LATTICE_NOTEADDRESS_WIDTH = 1(97D84A22) ),
```

See Also ▶"Using Programming File Utility" on page 1033

▶ "Running Programming File Utility" on page 1033

Securing the Device and Setting the Persistent Bit

The Security and Persistent Fields Editor allows you to read the settings of the security and the persistent bit of the bitstream file and change the setting to ON or OFF.

To read the settings of the security and persistent fields:

1. Choose **Tools > Security and Persistent Fields Editor.**
2. Click the **Browse** button and select the bitstream data file.
3. Click **Read** to read the settings for Security and Persistent Bit.

   The editor reads the settings and displays them in the Status cells.
4. To change a setting from OFF to ON or vice versa, click the status cell.
5. Do either of the following to save the changes:
   - To save the changes to the current data file, click **Write** (or click the arrow beside **Write** and then choose **Overwrite**). In the pop-up dialog box, click **Yes** to confirm that you want to overwrite the data file.
   - To save the changed data file to a different location or with a different file name, click the arrow beside **Write** and then choose **Save As**. In the Save As Data File dialog box, specify file name and location, and click **Save**.
6. A message box tells you that Write to file was successful. Click **OK**.
7. Click **Close** to exit the Security and Persistent Fields Editor.

See Also ▶"Using Programming File Utility" on page 1033

▶ "Running Programming File Utility" on page 1033

Programming File Utility Options

This section lists the options available in Programming File Utility. Topics include:

- "Feature Row Editor Dialgog Box" on page 1038
- "Control Register0 Editor Dialog Box" on page 1038
- "USERCODE Editor Dialog Box" on page 1038

Feature Row Editor Dialgog Box

The following options are available in the Feature Row Editor dialog box:

Device Lists the project's device. If a device is not displayed, use the browse button to locate the correct data file. Once a device is displayed, click **Read** to get the correct Max Bits/Digits information.

Read Reads the data file values and displays it in the display field.

Save Saves the data file.

Save As saves the data file with another file name.

Close Closes this dialog box.

See Also ▶"Programming File Utility Options" on page 1038

Control Register0 Editor Dialog Box

The following options are available in the Control Register0 Editor dialog box:

Device Lists the project's device. If a device is not displayed, use the browse button to locate the correct data file. Once a device is displayed, click **Read** to get the correct Max Bits/Digits information.

Read Reads the data file values and displays it in the display field.

Save Saves the data file.

Save As saves the data file with another file name.

Close Closes this dialog box.

See Also ▶"Programming File Utility Options" on page 1038

USERCODE Editor Dialog Box

The following options are available in the USERCODE Editor dialog box:

Device Lists the project's device. If a device is not displayed, use the browse button to locate the correct data file. Once a device is displayed, click **Read** to get the correct Max Bits/Digits information.

USERCODE Format Allows you to display the USERCODE field in hexadecimal, decimal, or ASCII format.

Usercode Allows you to specify a USERCODE value to override the current value in the data file.

Read Reads the USERCODE field in the data file and displays it in the display field.

Save Writes the value in the USERCODE display field and saves the data file.

Save As Writes the value in the USERCODE display field allows you to save the data file with another file name.

Close Closes this dialog box.

See Also ▶"Programming File Utility Options" on page 1038

Chapter 13

Testing and Debugging On-Chip

The final stage of developing a design is testing it on the actual FPGA on either a test board or in your system. Lattice Semiconductor offers two tools for debugging both the hardware aspect of the design and—if you are using the LatticeMico32 microprocessor—the software aspect:

- Reveal Analyzer to check for and to analyze specific events on signals
- LatticeMico Debugger to debug LatticeMico32 microprocessor software

Note

LatticeMico Debugger does not support the LatticeMico8 microcontroller.

While Reveal Analyzer and LatticeMico Debugger work independently, they can run at the same time.

Before you start debugging, each tool requires that a special interface module be added to the design. Then you rerun the design implementation process (Synthesize Design, Translate Design, Map Design, Place & Route Design) and generate bitstream data or a JEDEC file (depending on the device family) to program the FPGA. The tools can share the same ispDOWNLOAD cable and JTAG port that Programmer uses to download the design.

About Reveal Logic Analysis

One of the most common activities in debugging is logic analysis. To do this, use Reveal Inserter and Reveal Analyzer. You can use Reveal Inserter and Analyzer with all FPGAs and with MachXO and MachXO2 devices of 1200 or more LUTs.

Reveal continuously monitors signals within the FPGA for specific conditions, which can range from simple to quite complex. When the trigger condition occurs, Reveal can save signal values preceding, during, and following the event for analysis, including a waveform presentation. The data can be saved to a value change dump file (.vcd), which can be used with tools such as ModelSim, or to an ASCII tabular format that can be used with tools such as Excel.

To use Reveal Analyzer, first use Reveal Inserter to add Reveal modules to your design. In these modules, you specify the signals to monitor, define the trigger conditions, and other options. You can have up to 15 modules, typically one for each clock region of interest. When the modules are set up, regenerate the bitstream data or JEDEC file to program the FPGA.

Before starting a test run, set up Reveal Analyzer. It includes a number of options, including modifying trigger conditions and customizing the waveform display. You can save these settings for later use. During and after a test run, view the incoming data in Reveal's LA Waveform view. You can also save the data to a .vcd or .txt file to analyze with other tools.

See Also

- Reveal User Guide
- Reveal Troubleshooting Guide

Using the Reveal Example Project

If you want to get some hands-on experience with the Reveal tools, try using the example project while studying the online help.

The example is a simple 3-bit counter coded in Verilog. It already has a Reveal module inserted. It also already has trace data that can be viewed in Reveal Analyzer. A test board is not required to open the Reveal tools or to view the existing trace data. But, to actually run Reveal Analyzer to collect new data, you need a test board.

To start the Reveal example project:

1. Choose **File > Open > Design Example**.

   The Open Example dialog box opens.
2. Double-click **counta_reveal_XP2**.
3. Choose **count.ldf**.
4. Click **Open**.

The count design project opens. At this point, you can open Reveal Inserter.

Note

If you want to preserve the original example for future experiments, choose **File > Archive Project** now.

5. In the Processes view, double-click **Export Files** to implement the design.

   At this point, you can open Reveal Analyzer.

If you want to use the example project with a test board, change the device type to match the board by double-clicking the device in the File List view. Then run the design implementation process. See "Targeting a Device" on page 28.

Using the Reveal Debug Projects

If you are having trouble running Reveal with your design, Lattice provides the following pre-verified Reveal Debug Projects to allow you to verify that Reveal is working correctly on your computer.

- counter_impl_ECP2M
- counter_reveal_ECP2
- counter_reveal_ECP3
- counter_reveal_ECP5
- counter_reveal_LIFMD
- counter_reveal_MACHXO2
- counter_reveal_XO
- counter_reveal_XP2

The Reveal Debug Projects are located in a folder in the examples directory:

<diamond_install_path>\examples\reveal_debugger\

Your computer must be connected to a board with the appropriate Lattice device.

The reference design is a 32-bit counter which includes one input reset and 4 outputs LEDPIO_OUT0, LEDPIO_OUT1, LEDPIO_OUT2 and LEDPIO_OUT3.

Choose the appropriate device package and locate the output ports to see if the counter is toggling with the LED.

The clock signal is driven by internal oscillator and the OUTPOT_OUT3 to OUTPUTPIO_OUT0 are connecting to the most significant bit cnt[31:28]. This reference design has a Reveal module inserted and has trace signal TU1 that monitoring cnt[5:0] equal to 6b'100000 as trigger condition.

To start the Reveal example project:

1. Choose **File > Open > Project**.

   The Open Example dialog box opens.

2. Browse to the desired Reveal Debug Project folder, and choose the apppropriate **.ldf** file.
3. Click **Open**.
4. Ensure that the Diamond project settings match the device on your board.
5. Follow the steps outlined in "Performing Logic Analysis" on page 1073.
   - If you are able to run the Reveal Debug Project successfully in Reveal, then the problem may be with your design or in the clock source.
   - If you are unable to run the Reveal Debug Project successfully in Reveal, contact Lattice Technical Support as described in "Contacting Technical Support" on page 16, and send the project with the log files.

Debugging Software

If you include the LatticeMico32 microprocessor in your design, you will almost certainly be doing some on-chip software debugging. For this, use LatticeMico Debugger. With LatticeMico Debugger you can set breakpoints in the code, set watchpoints in the memory, inspect processor registers, and view many other aspects of the software and the processor.

When you're not sure if the problem is in the software or the hardware, use LatticeMico Debugger along with Reveal Analyzer.

To use LatticeMico Debugger, first use Mico System Builder (MSB) to add the debug module to the LatticeMico32 instance in your design. Then regenerate the bitstream data or JEDEC file to program the FPGA.

In C/C++ Software Project Environment (SPE), compile your software with debug symbols enabled. Then switch to the Debug perspective.

See Also

- Running LatticeMico System
- LatticeMico System Help
- LatticeMico32 Software Developer User Guide

Creating Reveal Modules

Create the modules for Reveal logic analysis with Reveal Inserter. The process basically consists of identifying trace signals, which are the signals that you want to analyze, defining a trigger signal, which is the event you want to analyze, and setting a few options. Then you insert the modules into your design and implement it.

With Reveal Inserter, it is easy to set up simple triggering conditions, as well as extremely complex ones. Triggering in Reveal is based on the trigger unit and the trigger expression. A trigger unit is used to compare signals to a value, and a trigger expression is used to combine trigger units to form a trigger signal.

Each Reveal Inserter project can include up to 15 modules. Each module has its own settings, trace signals, and trigger signal. In many cases, a single module is all that is required to debug a design. However, in designs with multiple clock regions, it may be necessary to sample different clock regions at the same time. For those types of designs, it is recommended that you use multiple modules, one for each clock region.

You can also set up modules that allow you to adjust and monitor serdes functions in ECP5UM designs.

Take some time to plan the logic analysis. Reveal modules can take a considerable amount of FPGA resources and creating the modules can take a significant amount of time. Find out how many EBRs and slices are still available. Consider adding all the trace signals and trigger events that you might want to analyze. The trigger signals can be modified in Reveal Analyzer while you're running tests if the necessary signals and capacity have been specified in the modules.

Note

Interactive and stand-alone synthesis are not compatible with Reveal modules. Reveal Inserter automatically uses the integrated synthesis option. Make sure your design project is set up for integrated synthesis. See "Pre-Synthesis Check List" on page 578.

About Reveal Inserter

Reveal Inserter has several views to help you manage the Reveal modules, find signals, and set up trace and trigger signals.

Dataset Dataset provides a list of all the modules in the Reveal project. To work on the trace and trigger signals of a module, select the module in the Dataset view. You can also add, remove, or rename modules by right-clicking

a module name and choosing from the drop-down menu. See “Managing the Modules in a Project” on page 1045.

Design Tree Design Tree provides a list of all the buses and signals in the design. You can select signals to trace and to trigger on by dragging them from the Design Tree view. To find signals, use the Signal Search function. See “Searching for Signals” on page 1047.

Design Tree also keeps track of how signals are being used. See “Viewing Signals in Design Tree” on page 1046.

Trigger Output Trigger Output provides a list of trigger signals available from other modules. These signals can also be traced and triggered on by the current logic analyzer module. Select these signals by dragging them from the Trigger Output view. To make a trigger available in the Trigger Output view, see “Trigger Out” on page 1068.

Trace Signal Setup Trace Signal Setup is where you assemble and organize the list of signals to be traced by the current logic analyzer module. Select signals by dragging them from the Design Tree and Trigger Output views. You can rearrange the signals in the list and organize them into groups. This view also provides several options for the trace operation. See “Setting Up Trace Signals” on page 1051.

Trigger Signal Setup Trigger Signal Setup is where you assemble the trigger for the current logic analyzer module. The trigger is the event that tells the module to save the data from the trace signals. Triggers are built up from “trigger units,” collections of signals compared to specific values, and “trigger expressions,” logical and sequential combinations of trigger units. See:

- “About Trigger Signals” on page 1056
- “Setting Up Trigger Units” on page 1057
- “Setting Up Trigger Expressions” on page 1060
- “Setting Trigger Options” on page 1067

Serdes Debug Setup Serdes Debug Setup is where you specify signals to control the current SERDES Debug module. Select signals by dragging them from the Design Tree view. See See “Creating SERDES Debug Modules” on page 1068.

Note

The Serdes Debug Setup tab is only visible when you select a Serdes Debug module in the Dataset pane.

Managing the Modules in a Project

Each Reveal Inserter project can include up to 15 modules. They are listed in the Dataset view. Working in the Dataset view or with the Debug menu, you can add, rename, and remove modules.

To add a module:

1. Choose **Debug > Add New Core**.
2. From the submenu, choose a module type.

   Reveal Inserter adds a new module in the Dataset pane.

To rename a module:

1. Double-click the module in the Dataset pane.
2. Type the new name of the module over the old name. The module name must:
   - Begin with a letter.
   - Consist of letters, numbers, and underscores (_).
   - Be different from all other modules in the Reveal project.
   - Be different from all other modules and instances in the design.
3. Press **Enter**.

To remove a module:

1. Select the module in the Dataset pane.
2. Press **Delete**.

Viewing Signals in Design Tree

The Design Tree pane shows the hierarchy of the whole design with all of the signals and buses. From the Design Tree pane, you can select signals and drag them to the Trace Signal Setup and Trigger Signal Setup views. To help you find signals, Design Tree has a search function, explained in "Searching for Signals" on page 1047.

The Design Tree pane also gives some information about the signals and how they are being used in the Reveal module. If you select a signal in the hierarchy, the Output tab displays information about it. If a signal is being used in the Reveal module, its name in the Design Tree pane is shown in bold type and followed by a symbol showing how the signal is being used:

- @Tc: trace signal
- @Tg: trigger unit signal
- @C: control signal (sample clock or sample enable)
- @Mx: mixed bus. Some of the signals are being used one way and some are not or are being used in another way.

To view all signals in the design hierarchy:

- Right-click on the design name in the Design Tree pane and choose **Expand All** from the pop-up menu.

To view the buses, ports, top-level signals, and top level of the hierarchy:

- Right-click on the design name in the Design Tree pane and choose **Collapse All** from the pop-up menu.

Searching for Signals

You can find signals in the Design Tree pane with a text search including wildcard characters.

To search for signals:

1. In the Signal Search box in the Design Tree pane, type the full name of the signal or a part of the name with wildcards (see the following table). The search is case-insensitive. The bit range of a bus, such as the "[7:0]" in cout[7:0] is not part of the name and should not be included in the search. Instead, just search on "cout".

Wildcard	Finds	Example
?	Any single character	"?out" finds aout, bout, and cout.
*	A sequence of any number of characters	"c*" finds cin and cout.
[<*string*>]	Any character in the string	"[ab]out" finds aout and bout.
[^<*string*>]	Any character except those in the string	"[^ab]out" finds cout but not aout or bout.
[<*c1*>-<*c2*>]	Any character in the range from <*c1*> through <*c2*>	"[a-c]out" finds aout, bout, and cout. "cout:[4-6]" finds cout:4, cout:5, and cout:6.
[^<*c1*>-<*c2*>]	Any character except those in the range from <*c1*> through <*c2*>	"cout:[^3-7]" finds cout:0, cout:1, and cout:2.

2. Click **Search**.

   If only one signal is found, it is highlighted it in the Design Tree pane.

   If more than one signal is found, the Search Result dialog box opens with a list of the signals.

3. If the Search Result dialog box opens, select the desired signals and click **OK**.

   To select one signal or bus, click on it. To select more than one, Control-click on each one. To select all signals and buses in a range, click on one end of the range and Shift-click the other end.

   The selected signals are highlighted in the Design Tree pane.

See Also "Viewing Signals in Design Tree" on page 1046

Checking the Design Rules

At any time you can check that your Reveal project is not violating any design rules such as proper module names, number of signals used, and required options set. The design rule check also tells you the total number of EBRs and slices needed for the project.

To check the Reveal module settings:

- Choose **Debug >** **Design Rule Check**.

  The results of the check are displayed in the Output tab.

Saving a Project

At any time you can save your Reveal project. Reveal Inserter automatically performs a design rule check when it saves a project. The results are shown in the Output tab.

Note

Reveal Inserter generates a new “signature,” or tracking mechanism, whenever a Reveal project is saved. Reveal Analyzer reads this signature to ensure that the FPGA has been programmed with the latest Reveal project. If you save the project without re-programming the FPGA, Reveal Analyzer will issue an error message, even if the Reveal project was not changed.

To save the project settings in the current directory:

- Choose **File >** **Save** <*filename*>.

To save the project settings in another directory:

- Choose **File > Save** <*filename*> **As**. In the Save Reveal Project dialog box, browse to the desired directory, enter the name of the .rvl file in the File Name box, and click **Save**.

See Also

- “Inserting the Reveal Modules” on page 1069

Logical Preference File Changes

Reveal Inserter modifies the logical preference (.lpf) file. These changes are handled automatically.

In the .lpf, Reveal:

- Modifies the timing settings needed for the debug logic.
- Adds an RVL_ALIAS preference, which maps clock names generated by Reveal to the clock names used in the original design. So you do not need to change your preferences that refer to those clock signals.

You may need to manually change preferences on some signals that Reveal Inserter is using. See “Preferences Not Recognized” on page 1070.

Limitations

Reveal Inserter has the following limitations:

Unsupported VHDL and Verilog Features in Reveal Inserter The following features that are valid in the VHDL and Verilog languages are not supported in Reveal Inserter:

- Array types of two dimensions or more are not shown in the port or node section.
- Undeclared wires attached to instantiated component instances are not shown in the hierarchical design tree. You must declare these wires explicitly if you want to trace or trigger with them.
- Variables used in conditional statements like if-then-else statements are not available for tracing and triggering.
- Variables used in selection statements like the case statement are not available for tracing and triggering.
- If function calls are used in the array declaration, the actual size of the array is unknown to Reveal Inserter.
- Entity and architecture of the same design cannot be in different files.
- In Verilog, you must explicitly declare variables at the very beginning of a module body to avoid obtaining different results from various synthesis tools.
- In VHDL, you must declare synthesis attributes within an entity, not within an architecture, to avoid obtaining different results from various synthesis tools.
- Signals used in VHDL “generate” statements are not available for tracing and triggering.
- Signals that are VHDL user-defined enumerated types, integer type, or Boolean type are not available for tracing and triggering.

Syn_keep and Preserve_signal Attributes In VHDL, always define the syn_keep and preserve_signal attributes as Boolean types when you declare them in your design. Synplify defines them as Boolean types, and Reveal Inserter will issue an error message if you define them as strings.

Signals Implemented as Hard Routes Signals that are implemented as hard routes in the FPGA instead of using the routing fabric are not available for tracing or triggering. Examples are connections to IB and OB components. Many common hard routes are automatically shown as unavailable in Reveal Inserter, but some are not. If you select a signal for tracing or triggering that is implemented as a hard route, an error will occur during the synthesis, mapping, placement, or routing steps.

Dangling or Unconnected Nets Dangling, or unconnected, nets in Verilog or VHDL code are available for use with Reveal Inserter. However, if you are using EDIF files as your source, dangling nets are probably not available because synthesis tools, which produce EDIF files, normally optimize out dangling logic.

Using Unlicensed IP

If your design contains an unlicensed IP block, the Hardtimer mechanism enables you to evaluate the IP. You can control this mechanism by choosing **Project > Active Strategy > Translate Design**, and setting the Hardware Evaluation option to Enable.

If you use the Reveal tools on a design that includes an unlicensed IP block, you cannot disable the Hardtimer mechanism. It is required to generate the bitstream data or JEDEC file for Reveal Analyzer. If you set the Hardware Evaluation option to Disable, the Reveal flow overwrites the Hardtimer mechanism. However, the option automatically reverts to Disable when you exit the Reveal flow.

Creating Logic Analysis Modules

Following are the major steps in creating a Logic Analysis module.

Before running Reveal Inserter the design must have been successfully synthesized.

Note

Reveal Inserter has limitations and requirements in how it works with the design. See "Limitations" on page 1049 and "Using Unlicensed IP" on page 1050.

To create a Logic Analyzer module:

1. After opening the design project, choose **Tools >**  **Reveal Inserter**.

   Reveal Inserter opens with the active Reveal project (.rvl) file. If there are no existing projects, Reveal Inserter creates one.

   Reveal Inserter also parses and statically elaborates the design. In some cases, code that was successfully synthesized is flagged as having an error. In these cases, Reveal Inserter is interpreting the HDL code more strictly than the chosen synthesis tool. It is likely that the code would not synthesize with a different synthesis tool or would have other compliance issues.

   To correct this problem, see the reveal_error.log file in the implementation directory. With the help of this file, locate and correct the problem in the code. Then synthesize the design and open Reveal Inserter again.

2. Select a module in the Dataset view. If you want to add another module to the project, choose **Debug > Add New Core > Add Logic Analyzer**. See also "Managing the Modules in a Project" on page 1045.

3. Set up the trace signals as desired. See “Setting Up Trace Signals” on page 1051.
4. Create trigger units, which are collections of signals and the values that will be part of causing a trigger signal. See “Setting Up Trigger Units” on page 1057.
5. Create trigger expressions, which are logical combinations of trigger units. See “Setting Up Trigger Expressions” on page 1060.
6. Select trigger options. See “Event Counter” on page 1067 and “Trigger Out” on page 1068.
7. When you have set up all your modules, add them to the design project by choosing **Debug > Insert Debug** and running the design implementation process. See “Inserting the Reveal Modules” on page 1069.

Setting Up Trace Signals

Setting up the trace signals for a module is mostly a matter of selecting the signals and buses that you want to watch and dragging them into the Trace Signal Setup tab. You can have up to 512 trace signals in a module.

The organization of the signals is up to you. Organize the signals according to how you want to view the data. Order them from top to bottom and organize them into buses. You are not limited to the bus organization of the design. You can completely rearrange the signals into your own organization of trace buses. You can manage the trace signals using the Debug menu or by right-clicking selected signals.

To set up trace signals:

1. Click the **Trace Signal Setup** tab.
2. Select a module in the Dataset view.
3. Select signals in the Design Tree and Trigger Output views and drag the signals to the desired position in the Trace Signal Setup tab. For help finding signals, see “Searching for Signals” on page 1047.

   Check the value to the right of the Implementation box to see how adding trace signals affects the consumption of FPGA resources.
4. You can further organize the signals in the Trace Signal Setup tab by ungrouping buses into individual signals and grouping signals into new buses.

   - To break a bus into individual signals, select the bus and choose **Debug > UnGroup Trace Bus**.
   - To create a new trace bus, select the desired signals and buses and choose **Debug > Group Trace Data**. Double-click the new bus and type in the desired name.

   You can also drag signals and buses into and out of other buses.

5. If you want to see what happens in the signals that define the trigger units, select **Include trigger signals in trace data** at the bottom of the Trace Signal Setup tab.

   A bus named “Trigger Signals” appears at the top of your list of trace signals. For details, see “Include trigger signals in trace data” on page 1052.

6. Set the trace options. There are several options that enhance the trace signals or control how data is sampled:

   - “Sample Clock” on page 1052
   - “Sample Enable” on page 1053
   - “Buffer Depth” on page 1054
   - “Timestamp” on page 1054
   - “Implementation” on page 1055
   - “Data Capture Mode” on page 1055
   - “POR Debug” on page 1055

See Also “Viewing Signals in Design Tree” on page 1046

Include trigger signals in trace data

You can add all the signals used in the trigger units to the trace signals. This allows you to see what happens on the signals that make up the trigger.

This option creates a bus named “Trigger Signals” at the top of your list of trace signals. Trigger Signals contains buses named for each trigger unit and containing the buses and signals used in the trigger units. The Trigger Signals bus cannot be moved or modified in any way.

To add the trigger signals to the trace signals:

- Select **Include trigger signals in trace data**.

Sample Clock

The sample clock determines when the trace signals are sampled. Reveal Analyzer samples the trace signals once every clock cycle on the clock’s rising edge.

To set the sample clock signal:

- Find the signal in the Design Tree view and drag it to the Sample Clock box of the Trace Signal Setup tab.

Note

On the board, make sure that the sample clock frequency is at least that of the JTAG clock. If the sample clock speed is too slow, you will be unable to complete capturing data with Reveal Analyzer.

The sample clock frequency should be no more than 200 MHz.

See Also ▶"Searching for Signals" on page 1047

Sample Enable

The sample enable is a signal that can be used to turn data capture on and off. Normally, data is captured for every sample clock cycle for a specified number of cycles. With sample enable, data capture only happens when the sample enable signal is active. Use sample enable to reduce the size of the trace buffer when there are stretches of data of no interest and these stretches are associated with a single signal.

An example is a design that contains many different sections but some sections only work during certain clock phases. The design uses a master clock and generates different signals for the phases. You could use one of the phase signals as the sample enable.

If the trigger occurs while the sample enable is inactive, Reveal Analyzer cannot accurately calculate the trigger point. Instead of showing the precise trigger point, Reveal Analyzer shows a trigger region that spans five clock cycles. Reveal Analyzer can guarantee that the trigger occurred in this region, but it cannot determine during which clock cycle the trigger occurred.

To set the sample enable:

1. Select **Sample Enable**.
2. Find the signal in the Design Tree view and drag it to the box in the Sample Enable section of the Trace Signal Setup tab.
3. In the box to the right of the signal name box, choose whether the signal is:

- **Active High**. Data can be captured when the sample enable is high.
- **Active Low**. Data can be captured when the sample enable is low.

See Also ▶"Searching for Signals" on page 1047

Buffer Depth

The buffer depth specifies the size of the trace memory buffer as the maximum number of samples that can be stored. The buffer should be deep enough to hold enough samples, before and after the trigger, for your analysis multiplied by the number of trigger events that you want to see in one test run. Available values are powers of two from 16 to 65,536.

For example, if you want 20 samples from each of five events, choose a buffer depth of at least 128.

To set the buffer depth:

1. Determine the maximum number of samples that you might want from a test run. Be generous to avoid re-implementing the design just to get a bigger buffer.
2. In the Buffer Depth box, choose a value that is at least as large as the desired number of samples.

Check the value to the right of the Implementation box to see how changing the buffer depth affects the consumption of FPGA resources.

Timestamp

The timestamp is a count of sample clock cycles from the beginning of a test run. This is different from the sample index that Reveal Analyzer automatically supplies. The sample index only provides a count of samples within each trigger's data set. The timestamp continues counting between triggers and when the sample enable signal blocks data capture.

However, unlike the sample index, the timestamp is extra data in each sample and so requires a larger trace buffer.

Use the timestamp when you want to know how long the test ran before triggering or how long the sample enable signal blocked data capture. The timestamp can also help associate triggers with external events or with data from another Reveal module using the same sample clock.

To add timestamps to the trace samples:

1. Select **Timestamp**.
2. Determine the number of sample clock cycles in the longest test run you might want to do.
3. In the drop-down menu to the right of "Timestamp," choose the size of the timestamp in bits. Choose the smallest value that can hold the count for the desired number of sample clock cycles.

   For example, if you might want to run a test for 50 thousand cycles, choose 16 bits.

Check the value to the right of the Implementation box to see how changing the timestamp size affects the consumption of FPGA resources.

Implementation

The implementation specifies what kind of RAM to use for the Reveal module. Normally you should choose EBR. But if you are short of EBR, choose distributed RAM.

To set the implementation:

- In the Implementation box, choose a kind of RAM:
  - **EBR** (embedded block RAM)
  - **DistRAM** (distributed RAM)

The number of EBR or slices needed is shown to the right. This value changes as you add or remove trace signals, or change the buffer depth or timestamp size.

Data Capture Mode

The data capture mode specifies whether Reveal Analyzer can look for one trigger or multiple triggers in a test run. Multiple Trigger Capture mode provides the greatest flexibility during test runs. Single Trigger Capture mode slightly reduces the amount of FPGA resources needed.

To set the data capture mode:

1. In the Data Capture Mode section, select one of the following:
   - **Single Trigger Capture**. Reveal Analyzer captures the data for only one trigger.
   - **Multiple Trigger Capture**. Reveal Analyzer captures the data for multiple triggers. The number of triggers is specified in Reveal Analyzer unless you are creating a module to monitor the power-on reset (POR) functions (see below).
2. If you select Multiple Trigger Capture, choose the “Minimum samples per trigger.” The number of samples collected for each trigger is set in Reveal Analyzer but cannot be smaller than this value.
3. If you select Multiple Trigger Capture and are creating a POR module, choose the “Number of triggers for POR.” For an explanation of POR modules, see “POR Debug” on page 1055.

POR Debug

To monitor power-on reset (POR) functions an automatic “trigger enable” signal must be built into the Reveal module. This is because POR functions happen immediately after power-on of the test board, before Reveal Analyzer can be started. When the trigger enable signal transitions to active, the module will watch for the trigger and collect samples. This is similar to clicking the Run  button in Reveal Analyzer. When Reveal Analyzer starts, it loads and displays any data collected by the POR modules.

POR modules are only supported with ECP5, LatticeECP3, LatticeSC, LatticeXP2, MachXO2, MachXO3D, and MachXO3L.

To set a POR trigger enable:

1. In the POR Debug section, select **Trigger Enable**.
2. Find the POR trigger signal in the Design Tree view and drag it to the text box in the POR Debug section.
3. Choose whether the signal is **Active High** or **Active Low**.

About Trigger Signals

Most of the trigger features need to be set up initially in Reveal Inserter but many of them can be modified in Reveal Analyzer while testing the design. See Table 136. Changes in Reveal Inserter mean the design has to be re-implemented (synthesis, map, place, and route) and reloaded into the FPGA. So it is worthwhile to be generous in defining your trigger units, trigger expressions, and other options. Think of setting up a trigger signal as creating capacity: access to signals and memory. Try to give yourself capacity to define all the triggering events that you might want to analyze later.

Table 136: Where Trigger Features Can Be Changed

Feature		Reveal Inserter	Reveal Analyzer
Trigger Units	Add	✔	
	Name	✔	✔
	Signals	✔	
	Operator	✔	✔
	Radix	✔	✔
	Value	✔	✔
Trigger Expressions	Add	✔	
	Remove	✔	✔
	Name	✔	✔
	Expression	✔	✔
	RAM type	✔	
	Maximum sequence depth	✔	
	Maximum event counter	✔	
Multiple Trigger Capture	Make available	✔	
	Number of samples per trigger	✔	✔
	Number of triggers		✔

Table 136: Where Trigger Features Can Be Changed

Feature		Reveal Inserter	Reveal Analyzer
Other Features	AND All versus OR All		✔
	Final event counter size	✔	✔
	Trace buffer depth	✔	
	Timestamp	✔	
	Trigger position		✔

See Also

- “Setting Up Trigger Units” on page 1057
- “Setting Up Trigger Expressions” on page 1060
- “Setting Trigger Options” on page 1067
- “Setting Up the Trigger Signals” on page 1080

Setting Up Trigger Units

The trigger unit is used to compare a number of signals to a value. A number of different operators are available for comparison and can be dynamically changed during analysis, along with the comparison value and the trigger unit name.

Each trigger unit can have up to 256 signals. Since there are 16 allowable trigger units, each module can have a maximum of 4096 trigger signals.

Try to minimize the number of trigger units in a module. The more trigger units there are, the longer it takes for Reveal Analyzer to configure the module at the beginning of a test run. More than eight trigger units may cause a delay of 30 seconds or more; 16 trigger units many cause a delay of 5 minutes.

Before you start setting up the trigger units, set the **Default Trigger Radix**. This option is in the Trigger Unit section of the Trigger Signal Setup tab. The default trigger radix is applied automatically to any new trigger units that you create but does not affect the radix of any existing trigger units. You can always change the radix of any trigger unit at any time.

To set up a trigger unit:

1. If you want to add a new trigger unit, click **Add** in the Trigger Unit section of the Trigger Signal Setup tab.
2. Specify the signals in the trigger unit by one of following methods:
   - Drag and drop signals from the Design Tree and Trigger Output views to the Signals column. For more information on using the Design Tree view, see “Viewing Signals in Design Tree” on page 1046 and “Searching for Signals” on page 1047.

- Double-click in the Signals column to open the TU Signals dialog box. See “Selecting and Ordering Trigger Signals” on page 1058.

3. If you want to change the order of the signals or remove any, double-click in the **Signals** column.

   The TU Signals dialog box opens. See “Selecting and Ordering Trigger Signals” on page 1058.

4. Click in the **Operator** column, and choose an operator from the drop-down menu. See “About Trigger Unit Operators” on page 1059.

   Both the operator type and the trigger unit value can be changed in Reveal Analyzer during hardware debugging.

5. If you want to change the radix of the Value column, click in the **Radix** column and choose a radix from the drop-down menu. The menu includes token sets whose bit width matches the trigger unit’s signals.

   Token sets are text labels for values that might appear on trace buses. You can create and apply token sets in Reveal Analyzer. See “Creating Token Sets” on page 1088.

6. Enter the comparison value in the **Value** column. The form of the value must match the specified radix. If you selected a token set in the Radix column, a drop-down menu opens in the Value column listing the tokens in the token set.

   You can use “x” for a don’t-care value if you selected binary, octal, or hexadecimal in the Radix column and if you selected ==, !=, or serial compare in the Operator column.

Selecting and Ordering Trigger Signals

In the Trigger Unit section of the Trigger Signal Setup view, double-click in the Signals column. The TU Signals dialog box opens.

To select and order trigger unit signals:

1. In the left box of the TU Signals dialog box, select signals that you want to use in the trigger unit. To select one signal or bus, click on it; to select more than one, Control-click on each one; to select all signals and buses in a range, click on one end of the range and Shift-click the other end.
2. Click **>** to add them to the box on the right.
3. To remove signals from the trigger unit, select them in the right box and click **<**.
4. Organize the signals by selecting one and clicking the up or down arrow until the signal is in its desired position. Continue until all the signals are in the desired order from least significant bit (LSB) down to the most significant bit (MSB).
5. Click **OK**.

About Trigger Unit Operators

Most of the trigger unit operators use standard logical comparisons between the current value of the combined signals of the trigger unit and a specified value. But some of the operators are unusual and need some explanation.

With the exception of “serial compare,” the operators can be changed in Reveal Analyzer.

Standard Logical Operators Reveal includes the following operators:

- == equal to
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to

Rising Edge and Falling Edge The “rising edge” and “falling edge” operators check for change in the signal value, not the value itself. So the trigger unit’s specified value is a bit mask showing which signals should have a rising or falling edge. A 1 means “look for the edge;” a 0 means “ignore this bit.” A multiple-bit value is true if any of the specified bits has the edge.

For example, consider a trigger unit defined as cout[3:0], rising edge, 1110. This trigger unit will be true only when cout[3], cout[2], or cout[1] have a rising edge. What happens on cout[0] does not matter.

- 0000 > 1110

  True because cout[3], cout[2], and cout[1] rose.

- 0000 > 1111

  True for the same reason. It does not matter whether cout[0] rises or not.

- 0000 > 0100

  True because a rising edge on any of the specified bits is sufficient.

- 1000 > 1000

  False because cout[3] did not rise. It just stayed high.

Serial Compare The “serial compare” operator checks for a series of values on a single signal. For example, if a trigger unit’s specified value is 1011, the “serial compare” operator looks for a 1 on the first clock, a 0 on the next clock, a 1 on the next clock, and a 1 on the last clock. Only after those four conditions are met in those four clock cycles is the trigger unit true.

Serial compare is available only when a single signal is listed in the trigger unit’s signal list. The radix is automatically binary.

You can only set the serial compare operator in Reveal Inserter. You cannot change it or select it in Reveal Analyzer as you can the other operators.

Managing Trigger Units

You can add and remove trigger units only in Reveal Inserter. You cannot add them in Reveal Analyzer.

To add a trigger unit:

- To add a new trigger unit, click **Add** in the Trigger Unit section of the Trigger Signal Setup tab.

To rename a trigger unit:

- Click in the Name column of the Trigger Unit section of the Trigger Signal Setup tab, and type in the new name. The name can consist of letters, numbers, and underscores. The first character must be either an underscore or a letter.

To remove a trigger unit:

1. In the Trigger Unit section of the Trigger Signal Setup tab, click in any box in the line representing the trigger unit that you want to remove.
2. Click **Remove**.

See Also ▶"Setting Up Trigger Units" on page 1057

Setting Up Trigger Expressions

You set up the initial trigger expressions in Reveal Inserter, but you can change them and their names in Reveal Analyzer. You can also enable or disable trigger expressions in Reveal Analyzer. However, you cannot change the maximum sequence depth or the maximum event counter of the trigger expressions in Reveal Analyzer.

To set up a trigger expression:

1. If you want to add a new trigger expression, click **Add** in the Trigger Expression section of the Trigger Signal Setup tab.
2. In the **Expression** column, enter the trigger expression. See "Trigger Expression Syntax" on page 1061.

   Reveal Inserter checks the syntax and displays the syntax in red font if it is erroneous.

   Both the trigger units and operators associated with a trigger expression can be changed in Reveal Analyzer during hardware debugging.
3. Click in the **Ram Type** column and choose whether the trigger expression is to be implemented with EBR (embedded block RAM) or slices (distributed RAM). The menu also shows how many of each resource would be needed. Normally you should choose EBR. But if you are short of EBR, choose slices.

4. Click in the **Max Sequence Depth** column and choose the maximum number of sequences, or trigger units connected by THEN operators, that can be used in the trigger expression.

   The maximum sequence depth must be at least as large as the number in the Sequence Depth column, which shows the number of sequences currently used by the trigger expression. Increasing the maximum sequence depth allows you to use more sequences if you change the trigger expression in Reveal Analyzer, but increasing the value also uses more FPGA resources. Consider the largest chain of sequences you might want to use in your test and choose a value at least that large. Reveal supports up to 16 sequence levels.

   The Max Sequence Depth value is set statically in Reveal Inserter and cannot be changed in Reveal Analyzer.

5. Click in the **Max Event Counter** box and choose the maximum size of the count in the trigger expression (the count is how many times a sequence must occur before a THEN statement).

   The Max Event Counter value must be at least as large as the largest counter value used in the trigger expression. Increasing the size of the event counter allows you to use larger counts if you change the trigger expression in Reveal Analyzer, but increasing the value also uses more FPGA resources. Consider the largest count you might want to use in your test and choose a value at least that large. The maximum is 65,536.

   You cannot change the Max Event Counter setting in Reveal Analyzer. You can only change it in Reveal Inserter.

See Also ▶"Example Trigger Expressions" on page 1065

Trigger Expression Syntax

Trigger expressions are combinations of trigger units. Trigger units can be combined in combinatorial, sequential, and mixed combinatorial and sequential patterns. A trigger expression can be dynamically changed at any time. Each module supports up to 16 trigger expressions that can be dynamically enabled or disabled in Reveal Analyzer. Trigger expressions support AND, OR, XOR, NOT, parentheses (for grouping), THEN, NEXT, # (count), and ## (consecutive count) operators. Each part of a trigger expression, called a sequence, can also be required to be valid a number of times before continuing to the next sequence in the trigger expression.

Detailed Trigger Expression Syntax Trigger expressions in both Reveal Inserter and Reveal Analyzer use the same syntax.

Operators You can use the following operators to connect trigger units:

- & (AND) – Combines trigger units using an AND operator.
- | (OR) – Combines trigger units using an OR operator.
- ^ (XOR) – Combines trigger units using a XOR operator.
- ! (NOT) – Combines a trigger unit with a NOT operator.
- Parentheses – Groups and orders trigger units.

- THEN – Creates a sequence of wait conditions. For example, the following statement:

  ```
  TU1 THEN TU2
  ```

  means “wait for TU1 to be true, then wait for TU2 to be true.”

  The following expression:

  ```
  (TU1 & TU2) THEN TU3
  ```

  means “wait for TU1 and TU2 to be true, then wait for TU3 to be true.”

  Reveal supports up to 16 sequence levels.

  See “Sequences and Counters” on page 1062 for more information on THEN statements.

- NEXT – Creates a sequence of wait conditions, like THEN, except the second trigger unit must come immediately after the first. That is, the second trigger unit must occur in the next clock cycle after the first trigger unit. See “Sequences and Counters” on page 1062 for more information on NEXT statements.
- # (count) – Inserts a counter into a sequence. See “Sequences and Counters” on page 1062 for information on counters.
- ## (consecutive count) – Inserts a counter into a sequence. Like # (count) except that the trigger units must come in consecutive clock cycles. That is, one trigger unit immediately after another with no delay between them. See “Sequences and Counters” on page 1062 for information on counters.

Case Sensitivity Trigger expressions are case-insensitive.

Spaces You can use spaces anywhere in a trigger expression.

Sequences and Counters Sequences are sequential states connected by THEN or NEXT operators. A counter counts how many times a state must occur before a THEN or NEXT statement or the end of the sequence. The maximum value of this count is determined by the Max Event Counter value. This value must be specified in Reveal Inserter and cannot be changed in Reveal Analyzer.

Here is an example of a trigger expression with a THEN operator:

```
TU1 THEN TU2
```

This trigger expression is interpreted as “wait for TU1 to be true, then wait for TU2 to be true.”

If the same example were written with a NEXT operator:

```
TU1 NEXT TU2
```

it is interpreted as “wait for TU1 to be true, then wait *one clock cycle* for TU2 to be true.” If TU2 is not true in the next clock cycle, the sequence fails and starts over, waiting for TU1 again.

The next trigger expression:

```
TU1 THEN TU2 #2
```

is interpreted as “wait for TU1 to be true, then wait for TU2 to be true for two sample clocks.” TU2 may be true on consecutive or non-consecutive sample clocks and still meet this condition.

The following statement:

```
TU1 ##5 THEN TU2
```

means that TU1 must occur for five consecutive sample clocks before TU2 is evaluated. If there are any extra delays between any of the five occurrences of TU1, the sequence fails and starts over.

The next expression:

```
(TU1 & TU2)#2 THEN TU3
```

means “wait for the second occurrence of TU1 and TU2 to be true, then wait for TU3.”

The last expression:

```
TU1 THEN (1)#200
```

means “wait for TU1 to be true, then wait for 200 sample clocks.” This expression is useful if you know that an event occurs a certain time after a condition.

You can only use one count (# or ##) operator per sequence. For example, the following statement is not valid, because it uses two counts in a sequence:

```
TU1 #5 & TU2 #2
```

Multiple count values are allowed for a single trigger expression, but only one per sequence. For two count operators to be valid in a trigger expression, the expression must contain at least one THEN or NEXT operator, as in the following example:

```
(TU1 & TU2) #5 THEN TU2 #2
```

This expression means “wait for TU1 and TU2 to be true for five sample clocks, then wait for TU2 to be true for two sample clocks.”

Also, the count operator must be applied to the entire sequence expression, as indicated by parentheses in the expression just given. The following is not allowed:

```
TU1 #5 & TU2 THEN TU2 #2
```

The count (#) operator cannot be used as part of a sequence following a NEXT operator. A consecutive count (##) operator may be used after a NEXT operator. The following is not allowed:

```
TU1 NEXT TU2 #2
```

The count (# or ##) operators can only be used in one of two areas:

- Immediately after a trigger unit or parentheses (). However, if the trigger unit is combined with another trigger unit without parentheses, a # cannot be used.
- After a closing parenthesis.

Precedence The symbols used in trigger expression syntax take the following precedence:

- Because it inserts a sequence, the THEN and NEXT operators always take the highest precedence in trigger expressions.
- Between THEN or NEXT statements, the order is defined by parentheses that you insert. For example, the following trigger expression:

  ```
  TU1 & (TU2|TU3)
  ```

  means “wait for either TU1 and TU2 or TU1 and TU3 to be true.”

  If you do not place any parentheses in the trigger expression, precedence is left to right until a THEN or NEXT statement is reached.

  For example, the following trigger expression:

  ```
  TU1 & TU2|TU3
  ```

  is interpreted as “wait for TU1 & TU2 to be true or wait for TU3 to be true.”
- The precedence of the ^ operator is same as that of the & operator and the | operator.
- The logic negation operator (!) has a higher precedence than the ^ operator, & operator, or | operator, for example:

  ```
  !TU1 & TU2
  ```

  means “not TU1 and TU2.”
- The # and ## operators have the same precedence as the ^ operator, & operator, or | operator. However, they can only be used in one of two areas:
  - Immediately after a trigger unit or trigger units combined in parentheses. However, if the trigger unit is combined with another trigger unit without parentheses, a # or ## operator cannot be used.

    Here is an example of correct syntax using the count (#) operator:

    ```
    TU1 #2 THEN TU3
    ```

    This statement means “wait for TU1 to be true for two sample clocks, then wait for TU3.”

    However, the following syntax is incorrect, because the count operator is applied to multiple trigger units combined without parentheses:

    ```
    TU1 & TU2#2 THEN TU3
    ```
  - After a closing parenthesis. Use parentheses to combine multiple trigger units and then apply a count, as in the following example:

    ```
    (TU1 & TU2)#2 THEN TU3
    ```

    This statement means “wait for the combination of TU1 and TU2 to be true for two sample clocks, then wait for TU3.”

See Also ▶"Example Trigger Expressions" on page 1065

Example Trigger Expressions

Following is a series of examples that demonstrate the flexibility of trigger expressions.

Example 1: Simplest Trigger Expression Following is the simplest trigger expression:

TU1

This trigger expression is true, causing a trigger to occur when the TU1 trigger unit is matched. The value and operator for the trigger unit is defined in the trigger unit, not in the trigger expression.

Example 2: Combinatorial Trigger Expression An example of a combinatorial trigger expression is as follows:

TU1 & TU2 | TU3

This trigger expression is true when (TU1 and TU2) or TU3 are matched. If no precedence ordering is specified, the order is left to right.

Example 3: Combinatorial Trigger Expression with Precedence Ordering In the following example of a combinatorial trigger expression, precedence makes a difference:

TU1 & (TU2 | TU3)

This trigger expression gives different results than the previous one. In this case, the trigger expression is true if (TU1 and TU2) or (TU1 and TU3) are matched.

Example 4: Simple Sequential Trigger Expression Following is an example of a simple sequential trigger expression:

TU1 THEN TU2

This trigger expression looks for a match of TU1, then waits for a match on TU2 a minimum of one sample clock later. Since this expression uses a THEN statement, it is considered to have multiple sequences. The first sequence is "TU1," since it must be matched first. The second sequence is "TU2," because it is only checked for a match after the first sequence has been found. The "sequence depth" is therefore 2.

The sequence depth is an important concept to understand for trigger expressions. Since the debug logic is inserted into the design, logic must be used to support the required sequence depth. Matching the depth to the entered expression can be used to minimize the logic. However, if you try to define a trigger expression that has a greater sequence depth than is available in the FPGA, an error will prevent the trigger expression from running. The dynamic capabilities of the trigger expression can therefore be

limited. To allow more flexibility, you can specify the maximum sequence depth when you set up the debug logic in Reveal Inserter. You can reserve more room for the trigger expression than is required for the trigger expression currently entered. If you specify multiple trigger expressions, each trigger expression can have its own maximum sequence depth.

Example 5: Mixed Combinatorial and Sequential Trigger Expression
Here is an example showing how you can mix combinatorial and sequential elements in a trigger expression:

TU1 & TU2 THEN TU3 THEN TU4 | TU5

This trigger expression only generates a trigger if (TU1 AND TU2) match, then TU3 matches, then (TU4 or TU5) match. You can set precedence for any sequence, but not across sequences. The expression (TU1 & TU2) | TU3 THEN TU4 is correct. The expression (TU1 & TU2 THEN TU3) | TU4 is invalid and is not allowed.

Example 6: Sequential Trigger Expression with Sequence Counts The next trigger expression shows two new features, the sequence count and a true operator to count sample clocks:

(TU1 & TU2)#2 THEN TU3 THEN TU4#5 THEN (1)#200

This trigger expression means wait for (TU1 and TU2) to be true two times, then wait for TU3 to be true, then wait for TU4 to be true five times, then wait 200 sample clocks. The count (# followed by number) operator can only be applied to a whole sequence, not part of a sequence. When the count operator is used in a sequence, the count may or may not be contiguous. The always true operator (1) can be used to wait or delay for a number of contiguous sample clocks. It is useful if you knew that an event that you wanted to capture occurred a certain time after a condition but you did not know the state of the trigger signals at that time.

However, there is a limitation on the maximum size of the counter. This depends on how much hardware is reserved for the sequence counter. When you define a trigger expression, the Max Event Counter setting in the Trigger Expression section of Reveal Inserter and Reveal Analyzer specifies how large a count value is allowed in the trigger expression. Each trigger expression can have a unique Max Event Counter setting.

See Also ▶"Trigger Expression Syntax" on page 1061

Managing Trigger Expressions

Trigger expressions are combinatorial or sequential equations of trigger units or both. Trigger expressions can be defined during insertion and changed in Reveal Analyzer. You can add up to 16 trigger expressions.

You can add trigger expressions only in Reveal Inserter. You cannot add them in Reveal Analyzer. You can dynamically enable or disable individual trigger expressions before triggering is activated during hardware debugging.

To add a trigger expression:

- In the Trigger Expression section of the Trigger Signal Setup tab, click **Add**.

To rename a trigger expression:

- Click in the Name column of the Trigger Expression section of the Trigger Signal Setup tab, and type in the new name. The name can consist of letters, numbers, and underscores. The first character must be either an underscore or a letter.

To remove a trigger expression:

1. Click in any box in the line representing the expression that you want to remove.
2. Click **Remove**.

See Also ▶Setting Up Trigger Expressions

Setting Trigger Options

In addition to the trigger units and trigger expressions, there are two other aspects of triggers: the final event counter and enabling the trigger signal for other Reveal modules.

Event Counter

The final event counter allows a counter to be added to the final trigger of one or more trigger expressions. In order to use the final event counter during logic analysis, you must specify it during insertion, along with the maximum count allowed. The actual count used by the counter during triggering can be dynamically changed during logic analysis.

To add a counter to the output of the final trigger:

1. Select **Enable final trigger counter** in the lower left portion of the Trigger Signal Setup tab.
2. In the **Event Counter Value** drop-down menu, choose the maximum size of the count of all the trigger expression outputs combined. You can choose powers of 2 between 2 and 65536.

   Clearing the “Enable final trigger counter” option is equivalent to setting the counter to a value of 1.

   You can change the value of this parameter in Reveal Analyzer, but you cannot make the value bigger, so be sure to reserve enough space for the count that you think you will need.

Trigger Out

To support triggers based on multiple sample clocks, cross-triggering is available between different debug modules. Reveal provides an optional trigger-out signal in the triggering section for every module.

If a design has multiple modules, trigger-out signals from other modules are listed as an available signal. To use a trigger-out signal as an input to another module, you must specify it as a NET or BOTH type. The IO type is only used for connecting the trigger-out to an external I/O. Trigger-out signals are listed in the Trigger Output view.

To create a trigger output signal:

1. Select **Enable Trigger Out**, at the bottom of the Trigger Signal Setup view.
2. In the Net drop-down menu, choose one of the following:
   - **IO** – Creates an I/O signal that can connect to an I/O pin.
   - **NET** – Creates a net signal that can be used in another module.
   - **BOTH** – Creates a signal that can be used another module and can connect to an I/O pin.
3. If you want to change the signal's name, double-click the name next to the Net menu and type in the new name.
4. In the Polarity menu, choose whether the signal is **Active High** or **Active Low**.
5. In the "Minimum pulse width" box, enter the minimum pulse width of the trigger output signal, measured in cycles of the sample clock. You can input any value of 0 or greater as the minimum pulse.
6. If you chose IO or BOTH for the Net type, go into the .lpf file and add a LOCATE preference to specify which pin the trigger-out should go to. See Applying Design Constraints. For example:

   LOCATE COMP "reveal_debug_count_LA0_net" SITE "J2" ;

Creating SERDES Debug Modules

SERDES Debug modules help in debugging the serdes function by giving you read and write access to control registers. In Reveal Analyzer you can monitor what is happening in the serdes and experiment with different control settings. These modules are only available with ECP5UM designs that use the DCU block.

To add a SERDES Debug module:

1. Choose **Debug > Add New Core > Add SERDES Debug**. See also "Managing the Modules in a Project" on page 1045.

   A new module appears in the Dataset view and the Serdes Debub Setup tab appears.

2. Select the sample clock in the Design Tree view and drag it to the Serdes Debug Setup tab. For help finding signals, see “Searching for Signals” on page 1047. For more about sample clocks, see “Sample Clock” on page 1052.
3. Select the serdes reset signal in the Design Tree view and drag it to the Serdes Debug Setup tab.
4. When you have set up all your modules, add them to the design project by choosing **Debug >** **Insert Debug** and running the design implementation process. See “Inserting the Reveal Modules” on page 1069.

Inserting the Reveal Modules

When you finish setting up the trace and trigger signals, you can insert the Reveal modules into the design.

Note

Interactive and stand-alone synthesis are not compatible with Reveal modules. Reveal Inserter automatically uses the integrated synthesis option. If you have not already, make sure your design project is set up for integrated synthesis. See Pre-Synthesis Check List.

To insert the debug logic modules into the design:

1. Choose **Debug >** **Insert Debug**.
2. In the Insert Debug to Design dialog box, select the modules to insert.
3. Select **Activate Reveal file in design project**.

   You should usually select this option. If the .rvl file is not active in the design project, the Reveal modules will not be included during synthesis.
4. Click **OK**.

   Reveal Inserter performs a design rule check and saves the Reveal (.rvl) file. The Output view shows resource requirements and the DRC report for the modules. The .rvl file is listed in the File List pane under Debug Files.
5. Implement the design in the usual way. See “Implementing the Design” on page 572.

   Adding Reveal modules can sometimes cause design implementation to fail. If you have problems, see “Troubleshooting Design Implementation Errors” on page 1069 and “Limitations” on page 1049.

Troubleshooting Design Implementation Errors

If the design implementation process fails after inserting Reveal modules, check this section for solutions. Some possible problems are preferences not being recognized and signals not being physically available for tracing or triggering.

Preferences Not Recognized When Reveal Inserter adds modules to a design, it must connect to the signals being used for triggers, traces, and the sample clock. If any of these signals have preferences on them, there is a chance that the preference may no longer be recognized. This only occurs on signals that do not go to the top level of the design when the Reveal module is not present. When the signal connects to a top-level Reveal module, the name on the net segment attached to the Reveal module is used for assigning preferences instead of the original name.

This problem occurs most often on the sample clock but other clocks may also be renamed. Check the clock section of the mapping report file (.mrp) for clock names. Being a clock signal, it is likely there is a preference already assigned to this signal. If the clock was not originally present in the top level of the design, the original preference is not recognized. This results in one or more warnings of the following form when Spreadsheet View is opened:

WARNING - baspe: Semantic Error: clocksig_200 matches no nets in the design. Occurred in "USE PRIMARY PURE NET "clocksig_200" ; ". Disabled this preference.

The solution is to copy the preference for the original signal and make a new preference for the net name that is now being used. Reveal Inserter always generates a name in the format of reveal_ist_<*number*>.

In the case of the above warning, the Period/Frequency sheet of Spreadsheet View shows a clknet named reveal_ist_67 instead of clocksig_200. In the preference file (.lpf), this can be corrected by copying the existing preference line ‘USE PRIMARY PURE NET “clocksig_200” ;’ and adding the line ‘USE PRIMARY PURE NET “reveal_ist_67” ;’.

By copying the preference instead of just changing it, the same preference file can be used both with and without the Reveal modules. If the debug connections are changed in Reveal Inserter, it may be necessary to change the preference due to Reveal Inserter generating a slightly different name.

Signals Implemented as Hard Routes Some signals that are implemented as hard routes in the FPGA instead of using the routing fabric are not available for tracing or triggering. Examples are connections to IB and OB components. If you select a signal for tracing or triggering that is implemented as a hard route, an error will occur during the synthesis, mapping, placement, or routing steps. However, the error can take many forms. Following are a couple of examples.

Here is an example error from the synthesis log file, <*cktname*>.log:

@E:"f:\cws\bugs\cr37986\reveal_workspace\tmprveal\rx_ddr_rvl.vhd":648:8:648:12|Port 'serin' on Chip 'RX_DDR' drives 1 PAD loads and 1 non PAD loads

In this example error message, the serin signal is a hard route, and serin is not the name of the original signal that was traced. The hierarchical path shown is for the Reveal module that was generated. It is not part of the original design and is not displayed during the debug module insertion. The

error message does not specify which trace or trigger signal is causing the problem.

To manually determine which signal is causing this error, you can use two approaches:

- Remove signals one by one in Reveal Inserter to see which caused the error. If you have only a few signals, this would be the best approach.
- Manually look through the design to determine the problem. If you have many signals, this approach would be the best.

  However, the error message refers to the temporary HDL design that is generated during debug logic insertion. Normally this HDL source is deleted after the database is built. To save this temporary HDL source in the case of errors in mapping, you must set an environment variable.

  a. In the System control panel, click on the Advanced tab, then click on the environment variable button at the bottom of the window.

  b. Create a new environment variable named KEEP_REVEAL_TEMP. The value can be anything, but it is normally set to TRUE.

  c. Once this variable is set, exit Diamond.

  d. Open Diamond and synthesize the design.

  You can now open the generated HDL to determine which signal caused the error. You can open this file with a text editor, or use HDL Explorer to open and explore the design. The top-level generated file is located at <*project_directory*>/reveal_workspace/tmpreveal/<*project_name*>_rvl.<*v or vhd*>.

Following is another example of an error generated during mapping:

ERROR - map: IO register/latch FF_inst cannot be implemented in PIC.

In this case, the input of a register is being traced. But the register is being implemented as an input flip-flop because of a preference, USE DIN. Allowing the register to be implemented as an internal flip-flop by removing the preference resolves the issue.

See Also ▶"Limitations" on page 1049

Removing Reveal Modules from the Design

When you want to remove the Reveal modules from your design, you can either remove the .rvl file from the design project or set it as inactive, leaving it easily available for future use. Otherwise, the modules will continue to be inserted.

To remove the Reveal modules from the design:

1. In the File List view, right-click the .rvl file.
2. Do one of the following:

- To remove the Reveal modules but keep the project, choose **Set as Inactive**.
- To delete the Reveal project, choose **Remove**.

3. Implement the design in the usual way. See Implementing the Design.

Performing Logic Analysis

Once you have created one or more Reveal modules in Reveal Inserter, you can use Reveal Analyzer to capture the trace signal data from your evaluation board and view that data as waveforms. As you run your tests and start getting results, you can modify the trigger units and expressions and the use of the trace buffer to test other conditions.

With ECP5UM designs you can also adjust serdes register settings and monitor serdes operation. See “Controlling Serdes” on page 1097.

To perform logic analysis:

1. If you will be working in a secured lab (without a network connection to the design project files), make a copy of the required files and install them in the lab computer. See “Working in a Secured Lab” on page 1073.
2. Connect your evaluation board and program the FPGA. See “Programming the FPGA” on page 1075.
3. Start Reveal Analyzer by selecting **Create a new file** in the Reveal Analyzer Startup Wizard. See “Starting Reveal Analyzer” on page 1076.
4. Set up the trigger signals for each module that you want to use. See “Setting Up the Trigger Signals” on page 1080.
5. Capture data. See “Capturing Data” on page 1089.
6. View the resulting waveforms for each module. See “Viewing Waveforms” on page 1091.
7. Optionally, you can save the data in a Reveal Analyzer (.rva) file, a value change dump (.vcd) file for use in third-party tools, or in an ASCII text (.txt) file. See “Saving the Reveal Analyzer Settings and Data” on page 1099.

To just view waveforms captured earlier:

1. Start Reveal Analyzer by selecting **Open an existing file** in the Reveal Analyzer Startup Wizard. See “Starting Reveal Analyzer” on page 1076.
2. View the waveforms. See “Viewing Waveforms” on page 1091.
3. Optionally, you can save the data in a Reveal Analyzer (.rva) file, a value change dump (.vcd) file for use in third-party tools, or in an ASCII text (.txt) file. See “Saving the Reveal Analyzer Settings and Data” on page 1099.

Working in a Secured Lab

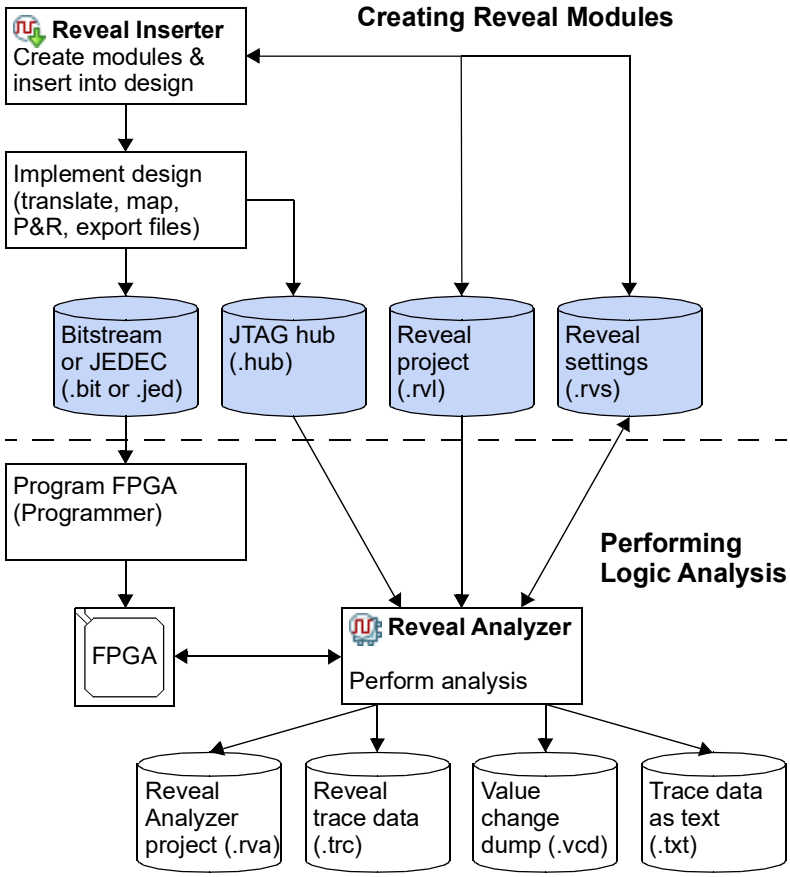
The usual process of Reveal logic analysis assumes that you are performing logic analysis on the same computer on which you created the Reveal modules, or that you are using a computer on the same network with easy access to the files. But sometimes your evaluation system is in a secured lab that is off the network.

In this situation use the stand-alone Reveal Analyzer and the stand-alone Programmer in the lab. To perform logic analysis, copy the files needed to

program the FPGA and to run Reveal Analyzer over to the lab computer. After you finish collecting trace data, you may want to copy settings or the trace data to bring them out of the lab.

The following figure shows the files needed by such a lab system and their sources. It also shows the files produced by Reveal Analyzer that you might want to bring out of the lab. See the following sections for more information about the files and how they are used.

Figure 29: Reveal Data Flow



Files to Bring into the Lab

The files you need to run Reveal Analyzer in the lab are:

- Bitstream (.bit) or JEDEC (.jed) file produced by the Export Files stage of the design implementation process. This file is required to program the FPGA.
- JTAG hub (.hub) file produced by the Translate Design stage of the design implementation process. This file is used by Reveal Analyzer to sort the data coming from the different Reveal modules.
- Reveal project (.rvl) file produced by Reveal Inserter. This is the defining file for a Reveal project and its modules. The .rvl file identifies the trace and trigger signals, and stores the trace and trigger options.

- Reveal settings (.rvs) file produced by Reveal Inserter. This file contains settings that can be changed in Reveal Analyzer while running tests. These settings include important parts of trigger units and trigger expressions.

Files to Bring Back

After collecting data with Reveal Analyzer, you may want to bring one or more files back from the lab either to update the Reveal project or to analyze the data somewhere else. See Figure 29 on page 1074.

If you made changes to the settings for trigger units or trigger expressions in Reveal Analyzer and want to update the project in Reveal Inserter with them, copy the .rvs file.

If you want to analyze the trace data outside the lab using Reveal Analyzer's LA Waveform view, choose **File >**  **Save** <*file*> and copy the following two files:

- Reveal Analyzer (.rva) file defines the Reveal Analyzer project. This file also contains data about the display of signals in the LA Waveform view.
- Reveal trace (.trc) file contains data recorded from the last test run. This file loads Reveal Analyzer's LA Waveform view.

If you want to analyze the trace data outside the lab using another tool, save the data as either a value change dump (.vcd) or text (.txt) file and copy that file. See "Saving to Other Formats" on page 1100.

Programming the FPGA

For an evaluation board using a single FPGA, program the FPGA in the usual way.

For an evaluation board using multiple FPGAs in a JTAG daisy chain, Reveal Analyzer has the following requirements:

- Only one device can be debugged at a time.
- The .xcf file must be in the design project directory.

To program an FPGA with a Reveal module in a daisy chain:

1. In Programmer, uncheck the Process column of the devices that do not get the Reveal module. This sets the operation of these devices to Bypass mode.
2. Ensure that the Process column of the device that does get the Reveal module is checked.
3. Double-click in the row of the device that does get the Reveal module.

   The Device Properties dialog box opens.

4. In the Device Properties dialog box, choose **Fast Program** from the drop-down menu.
5. Click **OK**.

For an FPGA that contains its own SPI flash, flash programming is also an option.

See Also

- "Programming the FPGA" on page 802

Starting Reveal Analyzer

Before starting Reveal Analyzer you need to decide if you want to work with a new Reveal Analyzer (.rva) file or an existing one. The .rva file defines the Reveal Analyzer project and contains data about the display of signals in the LA Waveform view. You may want to start Reveal Analyzer with a new file to set up a new test. Start with an existing file to rerun a test, to set up a new test based on existing settings, or to just view the waveforms from an earlier test.

How you start Reveal Analyzer also depends on whether you are using it integrated with Diamond or using the stand-alone version, and on your operating system.

Starting with a New File

Before you can start Reveal Analyzer with a new .rva file, you need to be connected to your evaluation board with a download cable and have the board's power turned on.

To start Reveal Analyzer with a new file:

1. Issue the start command. To start:
   - For integrated with Diamond, go to the Diamond main window and choose **Tools >** **Reveal Analyzer**.
   - For stand-alone in Windows, go to the Windows Start menu and choose **Programs > Lattice Diamond Reveal >** **Reveal Logic Analyzer**.
   - For stand-alone in Linux, go to a command line and enter the following:

     <*Reveal install path*>/bin/lin/**rvamain**

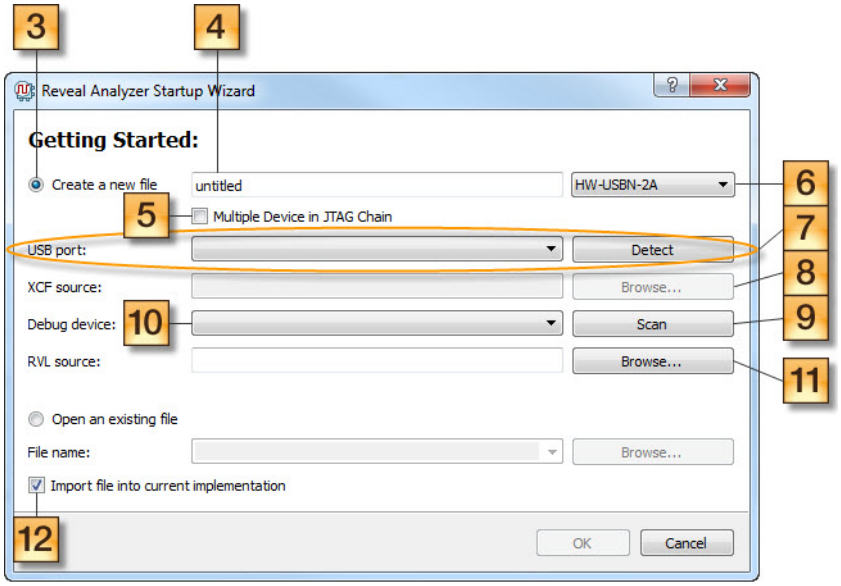
   If Reveal Analyzer finds just one .rva file in the active implementation, Reveal Analyzer opens with the data from that file. Otherwise, the Reveal Analyzer Startup Wizard dialog box appears.
2. If Reveal Analyzer opens with an existing file, choose **File > Save** <*file*> **As**.

The Save Reveal Analyzer File dialog box opens. Change the filename and click **Save**. You now have a new .rva file ready to work with.

3. In the Reveal Analyzer Startup Wizard dialog box, select **Create a new file** (at the upper-left of the dialog box).

   The dialog box presents a few rows of boxes that need to be filled in. In the figure below, the numbers match up with the steps in this procedure and show where the boxes, buttons, and menus are for each step.

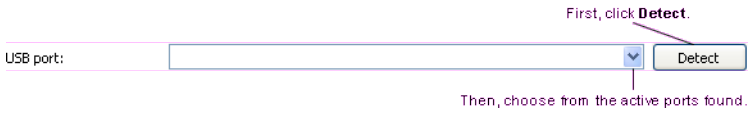
Figure 30: Startup Wizard with Steps for a New File



4. In the first row, type in the base name of the file. The extension is added automatically.
5. If there are daisy-chained devices, select **Multiple Device in JTAG Chain**.
6. To the right of this row is a drop-down menu. Choose the type of cable that your board is connected to.

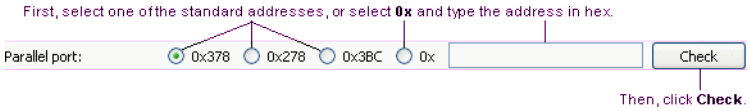
   Another row in the dialog box (circled in the figure above) changes to select the port.
7. Select the port. The method depends on the cable type:

   - If USB, click **Detect**. Then choose from the active ports found. The following figure shows the row after choosing a USB type.



   - If parallel, select the port address. If it's not one of the standard addresses given, select **0x** and type in the hexadecimal address.

Then click **Check** to verify that the connection is working. The following figure shows the row after choosing a parallel type.



8. If there are daisy-chained devices, click **Browse** in the XCF source row to find the XCF source file.
9. Click **Scan** to find the FPGA.
10. If there is more than one FPGA on your board, go to the “Debug device” menu and choose one that has a Reveal icon. The icon indicates the presence of a Reveal module.
11. Click **Browse** in the RVL source row to find the Reveal Inserter project (.rvl) file.
12. To add the new .rva file to the File List view, select **Import file into current implementation**. The .rva file works the same either way.
13. Click **OK**.

See Also

- “Creating a New Source File” on page 36

Starting with an Existing File

If you want to start with an existing file, you just need to have that .rva file in the design project. You need to be connected to the evaluation board only if you want to run a test and capture data.

To start Reveal Analyzer with an existing file:

1. Issue the start command. To start:
   - In the Diamond main window, choose **Tools > Reveal Analyzer**.
   - The stand-alone Reveal Analyzer in Windows, go to the Windows Start menu and choose **Programs > Lattice Diamond Reveal > Reveal Logic Analyzer**.
   - The stand-alone Reveal Analyzer in Linux, enter the following on a command line:

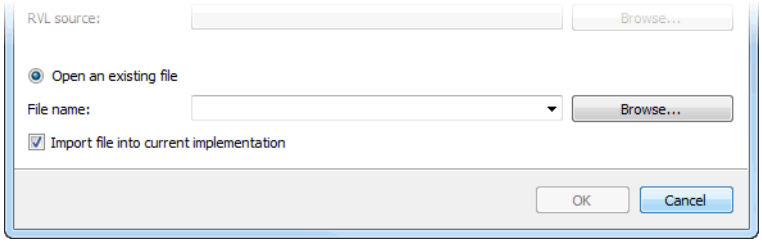
     <*Reveal install path*>/bin/lin/**rvamain**

   If Reveal Analyzer finds just one .rva file in the active implementation, Reveal Analyzer opens with the data from that file. Otherwise, the Reveal Analyzer Startup Wizard dialog box appears.

   If Reveal Analyzer opens with the .rva file you want to use, you’re ready to go. Otherwise continue with the following steps.

2. In the Reveal Analyzer Startup Wizard dialog box, select **Open an existing file** (in the lower part of the dialog box). See following figure.

Figure 31: Existing File Part of Startup Wizard



3. In the “File name” box, choose one of the available .rva files.
4. If the file you want is not in the menu, click **Browse** and browse to the desired .rva file.
5. To add the .rva file to the File List view, select **Import file into current implementation**. The .rva file works the same either way.
6. Click **OK**.

If the connection to your evaluation board has changed, either in the cable type or the computer port used, you need to tell Reveal Analyzer about the new connection. See “Changing the Cable Connection” on page 1079.

Changing the Cable Connection

If you need to change how your evaluation board is connected to your computer, go ahead and make the change. Then go through the following procedure to change the Reveal Analyzer project.

To change the cable setting in a Reveal Analyzer project:

1. Make sure your evaluation board is connected and that its power is on.
2. If Reveal Analyzer is not already open, start it as described in “Starting with an Existing File” on page 1078.
3. Choose **Design > Cable Connection Manager**.

   The Cable Connection Manager dialog box opens.
4. In the dialog box, choose the cable type.

   The second row in the dialog box changes to select the specific port.
5. Select the specific port. The method depends on the port type:
   - If USB, click **Detect**. Then choose from the active ports found.
   - If parallel, select the port address. If it’s not one of the standard addresses given, select **0x** and type in the hexadecimal address. Then click **Check** to verify that the connection is working.
6. To change the clock speed of the cable connection, adjust the value of TCK Low Pulse Width Delay.

7. Click **OK**.

Setting Up the Trigger Signals

In Reveal Analyzer, you cannot create new trigger units or trigger expressions, but you can change how they are named and defined. In trigger units, you can change the operators, radices, and values. In trigger expressions, you can change the expression including the operators and which trigger units are used. You can also modify some of the trigger options and the trigger position.

To modify the trigger of a module.

1. Click on the **LA Trigger** tab.
2. Choose the module from the drop-down menu in the Reveal Analyzer toolbar.

   The definition of the module's trigger units and trigger expressions appear. Grayed out cells and options cannot be changed in Reveal Analyzer. To change them, you must go back to Reveal Inserter.
3. Modify the trigger units as desired. If you want to change:
   - Operator: Choose an operator from the drop-down menu. You cannot change or select serial compare in Reveal Analyzer. See "About Trigger Unit Operators" on page 1081.
   - Radix of the value: Choose a radix from the drop-down menu. The menu includes token sets whose bit width matches the trigger unit's signals.
   - Value: Select it and type in a new comparison value. The form of the value must match the specified radix. If you selected a token set in the Radix column, a drop-down menu opens in the Value column listing the tokens in the token set.

     You can use "x" for a don't-care value if you selected binary, octal, or hexadecimal in the Radix column and if you selected ==, !=, or serial compare in the Operator column.
4. Modify the trigger expressions as desired. Trigger expressions can be completely rewritten provided that you do not exceed the maximum sequence depth or the maximum event counter. See "Trigger Expression Syntax" on page 1082.
5. In the Trigger Expression Name column, select the trigger expressions that you want to use in the next test.
6. In the Trigger Options section, at Enable TE, choose how to combine the selected trigger expressions to form the trigger event:
   - **AND All** means that all of the trigger expressions must be satisfied to form the trigger event.
   - **OR All** means that only one of the trigger expressions must be satisfied to form the trigger event.

7. If you want the trigger event, as specified in the Enable TE option, to happen more than once before capturing trace data, select **Final Event Counter**, in the Trigger Position section. Then specify the number of times the event is to happen to produce the actual trigger on the trigger signal. This option is available only if the event counter was enabled for the module in Reveal Inserter.
8. In the Trigger Options section, specify the number of samples per trigger and the number of triggers you want to record. The number of samples multiplied by the number of triggers cannot be greater than the trace buffer depth that was specified in Reveal Inserter. Reveal Analyzer adjusts the values when needed.
9. In the Trigger Position section, specify the trigger position relative to the trace data. The numbers in the section title show the current position. For example, “8/128” means the trigger happens on sample 8 out of the 128 samples per trigger. Select either:
   - **Pre-selected** and choose one of the standard positions:
     - Pre-Trigger: 1/16 of the way from the beginning of the samples.
     - Center-Trigger: 1/2 of the way from the beginning of the samples.
     - Post-Trigger: 15/16 of the way from the beginning of the samples.
   - **User-selected** and choose a position with the slider.

About Trigger Unit Operators

Most of the trigger unit operators use standard logical comparisons between the current value of the combined signals of the trigger unit and a specified value. But some of the operators are unusual and need some explanation.

With the exception of “serial compare,” the operators can be changed in Reveal Analyzer.

Standard Logical Operators Reveal includes the following operators:

- == equal to
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to

Rising Edge and Falling Edge The “rising edge” and “falling edge” operators check for change in the signal value, not the value itself. So the trigger unit’s specified value is a bit mask showing which signals should have a rising or falling edge. A 1 means “look for the edge;” a 0 means “ignore this bit.” A multiple-bit value is true if any of the specified bits has the edge.

For example, consider a trigger unit defined as cout[3:0], rising edge, 1110. This trigger unit will be true only when cout[3], cout[2], or cout[1] have a rising edge. What happens on cout[0] does not matter.

- 0000 > 1110

  True because cout[3], cout[2], and cout[1] rose.

- 0000 > 1111

  True for the same reason. It does not matter whether cout[0] rises or not.

- 0000 > 0100

  True because a rising edge on any of the specified bits is sufficient.

- 1000 > 1000

  False because cout[3] did not rise. It just stayed high.

Serial Compare The “serial compare” operator checks for a series of values on a single signal. For example, if a trigger unit’s specified value is 1011, the “serial compare” operator looks for a 1 on the first clock, a 0 on the next clock, a 1 on the next clock, and a 1 on the last clock. Only after those four conditions are met in those four clock cycles is the trigger unit true.

Serial compare is available only when a single signal is listed in the trigger unit’s signal list. The radix is automatically binary.

You can only set the serial compare operator in Reveal Inserter. You cannot change it or select it in Reveal Analyzer as you can the other operators.

Trigger Expression Syntax

Trigger expressions are combinations of trigger units. Trigger units can be combined in combinatorial, sequential, and mixed combinatorial and sequential patterns. A trigger expression can be dynamically changed at any time. Each module supports up to 16 trigger expressions that can be dynamically enabled or disabled in Reveal Analyzer. Trigger expressions support AND, OR, XOR, NOT, parentheses (for grouping), THEN, NEXT, # (count), and ## (consecutive count) operators. Each part of a trigger expression, called a sequence, can also be required to be valid a number of times before continuing to the next sequence in the trigger expression.

Detailed Trigger Expression Syntax Trigger expressions in both Reveal Inserter and Reveal Analyzer use the same syntax.

Operators You can use the following operators to connect trigger units:

- & (AND) – Combines trigger units using an AND operator.
- | (OR) – Combines trigger units using an OR operator.
- ^ (XOR) – Combines trigger units using a XOR operator.
- ! (NOT) – Combines a trigger unit with a NOT operator.
- Parentheses – Groups and orders trigger units.

- THEN – Creates a sequence of wait conditions. For example, the following statement:

  ```
  TU1 THEN TU2
  ```

  means “wait for TU1 to be true, then wait for TU2 to be true.”

  The following expression:

  ```
  (TU1 & TU2) THEN TU3
  ```

  means “wait for TU1 and TU2 to be true, then wait for TU3 to be true.”

  Reveal supports up to 16 sequence levels.

  See “Sequences and Counters” on page 1083 for more information on THEN statements.

- NEXT – Creates a sequence of wait conditions, like THEN, except the second trigger unit must come immediately after the first. That is, the second trigger unit must occur in the next clock cycle after the first trigger unit. See “Sequences and Counters” on page 1083 for more information on NEXT statements.
- # (count) – Inserts a counter into a sequence. See “Sequences and Counters” on page 1083 for information on counters.
- ## (consecutive count) – Inserts a counter into a sequence. Like # (count) except that the trigger units must come in consecutive clock cycles. That is, one trigger unit immediately after another with no delay between them. See “Sequences and Counters” on page 1083 for information on counters.

Case Sensitivity Trigger expressions are case-insensitive.

Spaces You can use spaces anywhere in a trigger expression.

Sequences and Counters Sequences are sequential states connected by THEN or NEXT operators. A counter counts how many times a state must occur before a THEN or NEXT statement or the end of the sequence. The maximum value of this count is determined by the Max Event Counter value. This value must be specified in Reveal Inserter and cannot be changed in Reveal Analyzer.

Here is an example of a trigger expression with a THEN operator:

```
TU1 THEN TU2
```

This trigger expression is interpreted as “wait for TU1 to be true, then wait for TU2 to be true.”

If the same example were written with a NEXT operator:

```
TU1 NEXT TU2
```

it is interpreted as “wait for TU1 to be true, then wait *one clock cycle* for TU2 to be true.” If TU2 is not true in the next clock cycle, the sequence fails and starts over, waiting for TU1 again.

The next trigger expression:

```
TU1 THEN TU2 #2
```

is interpreted as “wait for TU1 to be true, then wait for TU2 to be true for two sample clocks.” TU2 may be true on consecutive or non-consecutive sample clocks and still meet this condition.

The following statement:

```
TU1 ##5 THEN TU2
```

means that TU1 must occur for five consecutive sample clocks before TU2 is evaluated. If there are any extra delays between any of the five occurrences of TU1, the sequence fails and starts over.

The next expression:

```
(TU1 & TU2)#2 THEN TU3
```

means “wait for the second occurrence of TU1 and TU2 to be true, then wait for TU3.”

The last expression:

```
TU1 THEN (1)#200
```

means “wait for TU1 to be true, then wait for 200 sample clocks.” This expression is useful if you know that an event occurs a certain time after a condition.

You can only use one count (# or ##) operator per sequence. For example, the following statement is not valid, because it uses two counts in a sequence:

```
TU1 #5 & TU2 #2
```

Multiple count values are allowed for a single trigger expression, but only one per sequence. For two count operators to be valid in a trigger expression, the expression must contain at least one THEN or NEXT operator, as in the following example:

```
(TU1 & TU2) #5 THEN TU2 #2
```

This expression means “wait for TU1 and TU2 to be true for five sample clocks, then wait for TU2 to be true for two sample clocks.”

Also, the count operator must be applied to the entire sequence expression, as indicated by parentheses in the expression just given. The following is not allowed:

```
TU1 #5 & TU2 THEN TU2 #2
```

The count (#) operator cannot be used as part of a sequence following a NEXT operator. A consecutive count (##) operator may be used after a NEXT operator. The following is not allowed:

```
TU1 NEXT TU2 #2
```

The count (# or ##) operators can only be used in one of two areas:

- Immediately after a trigger unit or parentheses (). However, if the trigger unit is combined with another trigger unit without parentheses, a # cannot be used.
- After a closing parenthesis.

Precedence The symbols used in trigger expression syntax take the following precedence:

- Because it inserts a sequence, the THEN and NEXT operators always take the highest precedence in trigger expressions.
- Between THEN or NEXT statements, the order is defined by parentheses that you insert. For example, the following trigger expression:

  ```
  TU1 & (TU2|TU3)
  ```

  means “wait for either TU1 and TU2 or TU1 and TU3 to be true.”

  If you do not place any parentheses in the trigger expression, precedence is left to right until a THEN or NEXT statement is reached.

  For example, the following trigger expression:

  ```
  TU1 & TU2|TU3
  ```

  is interpreted as “wait for TU1 & TU2 to be true or wait for TU3 to be true.”
- The precedence of the ^ operator is same as that of the & operator and the | operator.
- The logic negation operator (!) has a higher precedence than the ^ operator, & operator, or | operator, for example:

  ```
  !TU1 & TU2
  ```

  means “not TU1 and TU2.”
- The # and ## operators have the same precedence as the ^ operator, & operator, or | operator. However, they can only be used in one of two areas:
  - Immediately after a trigger unit or trigger units combined in parentheses. However, if the trigger unit is combined with another trigger unit without parentheses, a # or ## operator cannot be used.

    Here is an example of correct syntax using the count (#) operator:

    ```
    TU1 #2 THEN TU3
    ```

    This statement means “wait for TU1 to be true for two sample clocks, then wait for TU3.”

    However, the following syntax is incorrect, because the count operator is applied to multiple trigger units combined without parentheses:

    ```
    TU1 & TU2#2 THEN TU3
    ```
  - After a closing parenthesis. Use parentheses to combine multiple trigger units and then apply a count, as in the following example:

    ```
    (TU1 & TU2)#2 THEN TU3
    ```

    This statement means “wait for the combination of TU1 and TU2 to be true for two sample clocks, then wait for TU3.”

See Also ▶"Example Trigger Expressions" on page 1086

Example Trigger Expressions

Following is a series of examples that demonstrate the flexibility of trigger expressions.

Example 1: Simplest Trigger Expression Following is the simplest trigger expression:

TU1

This trigger expression is true, causing a trigger to occur when the TU1 trigger unit is matched. The value and operator for the trigger unit is defined in the trigger unit, not in the trigger expression.

Example 2: Combinatorial Trigger Expression An example of a combinatorial trigger expression is as follows:

TU1 & TU2 | TU3

This trigger expression is true when (TU1 and TU2) or TU3 are matched. If no precedence ordering is specified, the order is left to right.

Example 3: Combinatorial Trigger Expression with Precedence Ordering In the following example of a combinatorial trigger expression, precedence makes a difference:

TU1 & (TU2 | TU3)

This trigger expression gives different results than the previous one. In this case, the trigger expression is true if (TU1 and TU2) or (TU1 and TU3) are matched.

Example 4: Simple Sequential Trigger Expression Following is an example of a simple sequential trigger expression:

TU1 THEN TU2

This trigger expression looks for a match of TU1, then waits for a match on TU2 a minimum of one sample clock later. Since this expression uses a THEN statement, it is considered to have multiple sequences. The first sequence is "TU1," since it must be matched first. The second sequence is "TU2," because it is only checked for a match after the first sequence has been found. The "sequence depth" is therefore 2.

The sequence depth is an important concept to understand for trigger expressions. Since the debug logic is inserted into the design, logic must be used to support the required sequence depth. Matching the depth to the entered expression can be used to minimize the logic. However, if you try to define a trigger expression that has a greater sequence depth than is available in the FPGA, an error will prevent the trigger expression from

running. The dynamic capabilities of the trigger expression can therefore be limited. To allow more flexibility, you can specify the maximum sequence depth when you set up the debug logic in Reveal Inserter. You can reserve more room for the trigger expression than is required for the trigger expression currently entered. If you specify multiple trigger expressions, each trigger expression can have its own maximum sequence depth.

Example 5: Mixed Combinatorial and Sequential Trigger Expression
Here is an example showing how you can mix combinatorial and sequential elements in a trigger expression:

TU1 & TU2 THEN TU3 THEN TU4 | TU5

This trigger expression only generates a trigger if (TU1 AND TU2) match, then TU3 matches, then (TU4 or TU5) match. You can set precedence for any sequence, but not across sequences. The expression (TU1 & TU2) | TU3 THEN TU4 is correct. The expression (TU1 & TU2 THEN TU3) | TU4 is invalid and is not allowed.

Example 6: Sequential Trigger Expression with Sequence Counts The next trigger expression shows two new features, the sequence count and a true operator to count sample clocks:

(TU1 & TU2)#2 THEN TU3 THEN TU4#5 THEN (1)#200

This trigger expression means wait for (TU1 and TU2) to be true two times, then wait for TU3 to be true, then wait for TU4 to be true five times, then wait 200 sample clocks. The count (# followed by number) operator can only be applied to a whole sequence, not part of a sequence. When the count operator is used in a sequence, the count may or may not be contiguous. The always true operator (1) can be used to wait or delay for a number of contiguous sample clocks. It is useful if you knew that an event that you wanted to capture occurred a certain time after a condition but you did not know the state of the trigger signals at that time.

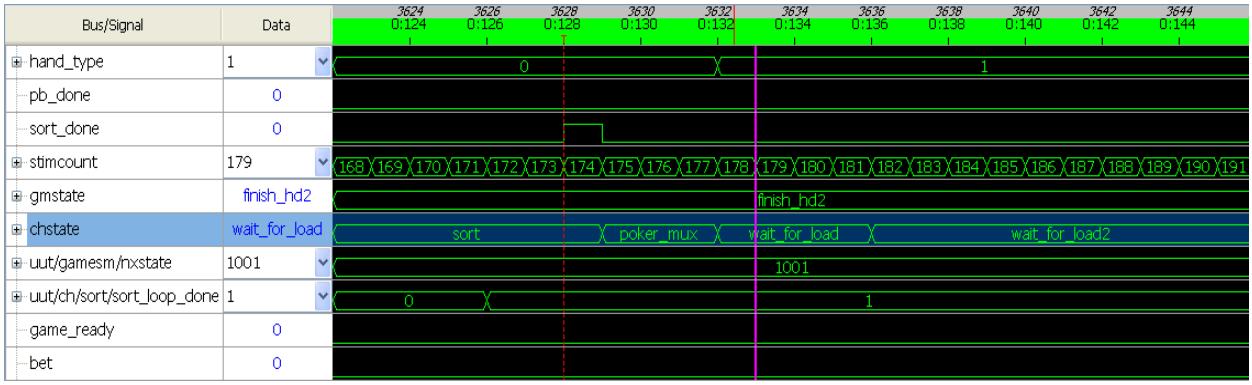
However, there is a limitation on the maximum size of the counter. This depends on how much hardware is reserved for the sequence counter. When you define a trigger expression, the Max Event Counter setting in the Trigger Expression section of Reveal Inserter and Reveal Analyzer specifies how large a count value is allowed in the trigger expression. Each trigger expression can have a unique Max Event Counter setting.

See Also ▶“Trigger Expression Syntax” on page 1082

Creating Token Sets

You can create sets of “tokens,” or text labels, for values that might appear on trace buses. You can create tokens such as ONE, TWO, THREE, or Reset, Boot, Load. Tokens can make reading the waveforms in Reveal Analyzer easier and can highlight the occurrence of key values. See the following figure for an example. The row for the chstate bus uses tokens.

Figure 32: LA Waveform View Using Tokens



To create or modify a token set:

1. Choose **Design > Token Set Manager**.

   The Token Manager dialog box opens. If the Reveal project already has token sets defined, they are listed in the dialog box.

2. If you want to use token sets that were previously saved to a separate file, right-click in the dialog box and choose **Import**. In the Import Tokens dialog box, browse to the token (.rvt) file and click **Open**.

   The token sets in the .rvt file are added to the list in Token Manager.

3. To create a new token set, click **Add Set**.

   A new token set is started with default values. But it has no tokens defined yet.

4. To change the size of the token values, double-click the value in the Num. of Bits column and type in the new width, in bits. The width can be up to 256. The width must be the same as the bus that the token set will be used with.

   The Num. of Bits value can only be changed when the token set is empty. If there are any tokens, you will get an error message.

5. To create a new token, select a token set. Then click **Add Token**.

   A new token is created with default values. Repeat for as many new tokens needed.

6. You can modify token sets by doing any of the following:

   - To change the name of a token or token set, double-click the name and type a new name. The name can consist of letters, numbers, and underscores (_). It must start with a letter.

- To change the value of a token, double-click the value and type in a new value. Token values must be prefixed by one of the radix indicators shown in the following table:

Radix	Prefix	Example
Binary	b’	b’110x0
Octal	o’	o’53
Decimal	d’	d’123
Hexadecimal	h’	x’0F2

  If a value does not have a prefix, its radix is assumed to be binary. You can use an “x” in binary numbers as a don’t-care value.

- To remove a token or token set, select it. Then click **Remove**.

7. You can save the collection of token sets showing in the dialog box to a separate file for use in another project. To save the token sets, right-click and choose **Export**. In the Export Tokens dialog box, browse to the desired location and type in the name of the new token (.rvt) file. Click **Save**.
8. When you are done, click **Close** to close the dialog box. The token sets are automatically applied to the current Reveal project.

Capturing Data

After you have configured trigger settings in the LA Trigger tab, you can capture data.

Before capturing data, your evaluation board must be connected and the design downloaded. See “Programming the FPGA” on page 1075.

To capture data:

1. In the Reveal Analyzer toolbar, select the modules you want to use.
2. Click the Run button in the Reveal Analyzer toolbar.

   The Run button changes into the Stop button and the status bar next to the button shows the progress.

Reveal Analyzer first configures the modules selected for the correct trigger condition, then waits for the trigger conditions to occur. When a trigger occurs, the data is uploaded to your computer. The resulting waveforms appear in the LA Waveform tab.

If the trigger condition is not met, Reveal Analyzer continues running. In that case, you can use manual triggering, described in “Using Manual Triggering” on page 1090.

See Also ▶"Stopping Data Capture" on page 1090

Stopping Data Capture

You can stop capturing data at any time.

To stop data capture:

1. Choose a module from the drop-down menu in the Reveal Analyzer tool bar.
2. Click the Stop button in the Reveal Analyzer toolbar.

This command only stops the data capture for the current module. You must stop each module separately.

Using Manual Triggering

If triggering fails to occur or you want to trigger manually instead of triggering when a signal condition occurs, you can use manual triggering to collect data. The data may then help you find out why triggering did not occur as you originally intended.

When you select manual triggering, Reveal Analyzer fills the buffer with data captured from that moment. In single-trigger capture mode, it fills the buffer and stops. In multiple-trigger capture mode, it captures one trigger and data. You can then continue to manually trigger as many times as the original triggering setup specified. If you want to capture fewer triggers, you can manually trigger the desired number of times, then click the Stop button. Then the buffer starts uploading the data.

To use manual triggering:

1. After you start capturing data with the Run button, choose a module from the drop-down menu in the Reveal Analyzer tool bar.
2. Click the Manual Trigger button.

   This command only applies to the data capture on the current module. You must start each module separately.
3. When you have captured the desired number of triggers in multiple trigger capture mode, click the Stop button.

Common Error Conditions

Reveal Analyzer may fail to generate waveforms and instead issue an error message because of problems with the inserted modules.

Clock Causes Module to Malfunction If Reveal Analyzer issues the following error message, the module may not be functional because of a clock problem:

```
Incorrect HUB ID. Check the clock or cable setup.
```

In this case, be sure the clock is running on the target hardware.

Signature in .rvl File Does Not Match Signature in Bitstream If Reveal Analyzer issues an error message similar to the following, the module may be functional, but the signature in the .rvl file does not match the signature in the bitstream:

ERROR: core0 incorrect signature
Problem cause is device programmed with incorrect (older) bitstream.

This problem can be caused by saving the Reveal Inserter project file, even if you have made no changes, without re-implementing the design.

If you receive this message, regenerate the bitstream or JEDEC file, including running all implementation stages, or download another file whose signature matches that of the .rvl file.

Sample Clock Runs Too Slowly The following module signature error message indicates that the sample clock is running too slowly:

ERROR: core0 incorrect signature
Problem cause is sample clock not running or running too slow.

On the board, make sure that the minimum sample clock frequency is at least three times (and preferably four times) that of the JTAG clock. If the sample clock speed is too slow, you will not be able to complete capturing data.

If the device signature is all ones:

ERROR: core0 incorrect signature (RVL File [1906089203] does not match Device [1111111111]).

there may have been a problem during implementation. The post-map netlist needs to be viewed directly to determine the root cause. Contact Lattice Technical Support.

Viewing Waveforms

After capturing data, you can view the trace data in waveform format in the LA Waveform tab. Whenever the trace stops, Reveal Analyzer reads the trace samples and automatically updates the signal waveforms.

To view a waveform:

1. Click the **LA Waveform** tab.
2. Choose a module from the drop-down menu in the Reveal Analyzer tool bar.

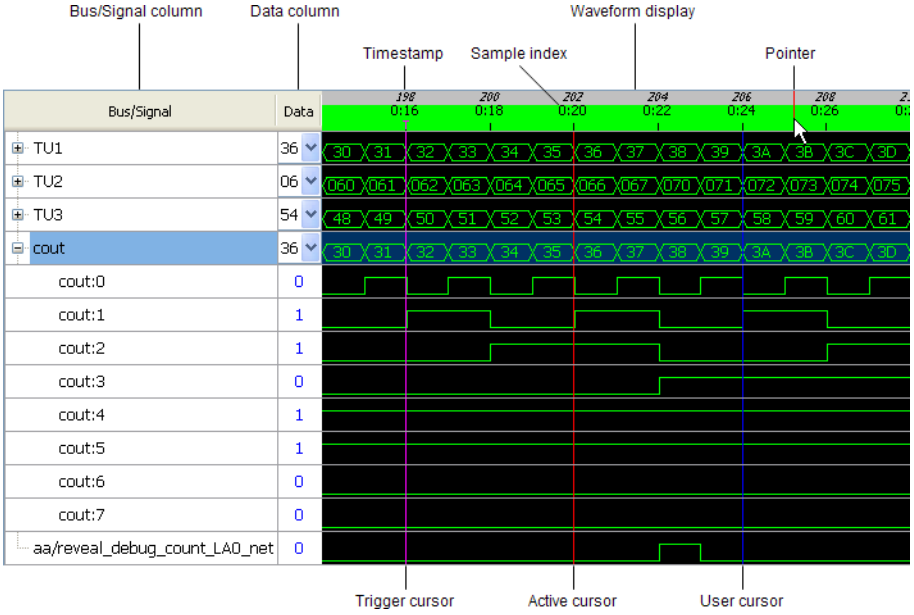
Sometimes the waveform includes fewer clock cycles than you expect; in particular, fewer before the trigger. This happens if the trigger occurs so

quickly after the test starts that there are not enough clock cycles before the trigger to fill that part of the trace buffer.

About the LA Waveform View

Waveforms are presented in a grid layout as shown in Figure 33 along with several features to help you find and analyze the data.

Figure 33: Elements of the LA Waveform View



Bus/Signal Column Displays the names of the trace buses and signals in the selected module.

Data Column Displays the value of the bus or signal at the active cursor (a solid, red line that you can set in the waveform display). Buses also have a drop-down menu for setting the radix used in the Data column and in the waveform display. The menu includes token sets whose bit width matches the bus. See “Setting a Trace Bus Radix” on page 1096.

Waveform Display Displays the trace data in waveform format. When there is room, bus values are included the display using the radix set in the Data column. You can zoom in and out, pan, and jump to various points.

The waveform display includes several other elements to help you read the display and analyze the data:

- Timestamp. The gray bar at the top of the display shows “timestamps” of the trace frames (actually, a simple count of the clock cycles). Timestamps are shown only if the Timestamp trace option was selected for the module in Reveal Inserter. See “Timestamp” on page 1054.
- Sample Index. The green bar near the top of the display shows a count of triggers and trace samples within each trigger’s data set. The sample indexes have the form <*trigger*>:<*sample*>. For example, 0:2 indicates the

first trigger and the third trace sample for that trigger (the counts are zero-based). 2:10 indicates the third trigger and the eleventh trace sample for that trigger.

- Pointer. A red line that cuts across the timestamp and sample index bars, the pointer follows the horizontal movement of the mouse pointer across the waveform display. Use the pointer to see where you are in time as you examine the waveform.
- Cursors. Vertical lines cutting through all the signals, cursors mark moments in the waveform. See “About Cursors” on page 1093.

See Also

- “Zooming In and Out” on page 1094
- “Moving around the Waveform” on page 1095

About Cursors

The LA Waveform view comes with three types of “cursors” to highlight moments in the waveform. The cursors are vertical lines cutting through all the signals at the leading edge of a clock cycle. See Figure 33 on page 1092. The three types are:

- Trigger. A purple line with a “T” at the top, trigger cursors are automatically placed at the moment of each final trigger event. If the module used a sample enable signal and the exact moment of the trigger is unknown, the waveform shows a trigger cursor five clock cycles before the sample enable signal turned inactive and sampling stopped.
- Active. A red line appears wherever you click in the waveform. The Data column shows the values of the signals and buses at the moment highlighted by the active cursor.
- User. A blue line can be placed anywhere you want. Use these cursors to mark moments of interest. You can also use these cursors to maneuver about a long waveform with the Go to Cursor command. See “Working with User Cursors” on page 1093.

Working with User Cursors

You can create any number of user cursors. They can be moved or deleted. You can also jump the display to any one of them.

Most cursor functions require that the LA Waveform view be in Select mode: right-click in the LA Waveform view and choose **Select Mode**.

To create a user cursor:

1. Click in the desired clock cycle.

   The active cursor appears. Make sure it is where you want the user cursor to be.

2. Right-click and choose **Add Cursor**.

To move a user cursor:

1. Zoom in so you can easily see and click in individual samples.
2. Click in the desired location.

   The active cursor appears. Make sure it is where you want the user cursor to be.
3. Carefully click in the sample to the right of the user cursor.

   You must click on or to the right of the user cursor. Otherwise you are just moving the active cursor to a neighboring sample.

   The user cursor and the active cursor exchange locations.

To jump to a user cursor:

- Right-click in the waveform and choose **Go to Cursor >** <*cursor*>. Cursors are identified by the sample index as shown in the green bar at the top of the waveform display.

To remove a user cursor:

1. Click on or near the cursor.

   The active cursor appears. Make sure it is on or next to the user cursor you want to remove.
2. Right-click and choose **Remove Cursor**.

To remove all user cursors:

- Right-click in the waveform and choose **Clear All Cursor**.

See Also "Moving around the Waveform" on page 1095

Zooming In and Out

You can zoom in, to expand the waveform and see more detail, and zoom out, to see more of the waveform.

To zoom in on a waveform:

- Choose **View > Zoom In**.

To zoom in on a specified area:

1. Right-click the waveform and choose **Zoom Mode**.

   The pointer changes to a cross: +.
2. Hold down the left mouse button and drag the pointer across the area you want to zoom in on.

   A shaded area appears on the waveform display.
3. Release the mouse button.

The shaded area expands to fill the display.

To zoom out on a waveform:

- Choose **View > Zoom Out**.

To show the entire waveform in the window:

- Choose **View > Zoom Fit**.

See Also "Moving around the Waveform" on page 1095

Moving around the Waveform

The waveform is usually much wider than the display, especially if you zoom in enough to see individual trace samples. To see nearby sections of the waveform, you can slide it left and right by panning (described below) or by using the horizontal scroll bar at the bottom of the LA Waveform view. You can also jump to various points in the waveform including any cursors you've placed, the trigger point, the start of the display, and the end.

To pan the waveform display:

1. Right-click in the waveform and choose **Pan Mode**.
2. Press the left mouse button and drag to the left or the right.

To jump to a location in the waveform:

- Right-click in the waveform and choose from the menu. To jump to the:
  - User cursor, choose **Go to Cursor >** <*cursor*>. Cursors are identified by the sample index as shown in the green bar at the top of the waveform display.
  - Trigger point, choose **Zoom > Zoom Trigger**
  - Start of the display, choose **Zoom > Zoom Start**
  - End of the display, choose **Zoom > Zoom End**

  The display changes to show the selection. The zoom level may change to keep the display filled.

See Also

- "About Cursors" on page 1093
- "Working with User Cursors" on page 1093
- "Zooming In and Out" on page 1094

Setting a Trace Bus Radix

You can set the radix of a trace bus displayed in the LA Waveform tab. You can choose a binary, octal, decimal, or hexadecimal radix. You can also use any token set whose bit width matches the bus.

To set the bus radix of a signal or bus:

1. In the LA Waveform tab, click in the Data cell of the signal or bus.

   A menu appears showing the different radices and any token sets that fit.

2. Choose the desired radix or token set.

To set the bus radix of multiple signals and buses:

This method can set several signals to the same radix but cannot use tokens.

1. In the LA Waveform tab, select one or more buses.

   To select one bus, click on it. To select more than one, Control-click on each one. To select all buses in a range, click on one end of the range and Shift-click the other end. If you want to change all the signals in the waveform to the same radix, you do not need to select anything.

2. Right-click in one of the selected waveforms and choose **Set Bus Radix**. Be careful to click in the same row as one of your selections, or you will change the selection.

   The Set Bus Radix dialog box opens.

3. In the drop-down menu, choose the radix.
4. In the Range drop-down menu, choose **Selected signals** or, if you want to change all the signals in the waveform to the same radix, choose **All signals**.
5. Click **OK**.

Changing LA Waveform Colors

You can change the colors used by the LA Waveform view.

To change the colors:

1. Choose **Tools > Options**.
2. In the Options dialog box, choose **Reveal Analyzer > Colors**.
3. Click on the color sample for the desired part of the LA Waveform view.

   The Select Color dialog box opens.

4. Select a color.
5. In the Select Color dialog box, click **OK**.
6. To see the effect of the change, click **Apply**.
7. Change other colors if desired.
8. Click **OK**.

Counting Samples

You can easily count the number of samples in a range on the display.

To count samples:

- Click where you want to start counting and drag to the end of the range.

  While you're dragging, the LA Waveform view shows two red lines and the number of samples between the lines.

Controlling Serdes

Reveal Analyzer provides two methods for controlling serdes functions in an ECP5UM device. You can use:

- A Serdes Debug module created with Reveal Inserter to experiment with control registers and monitor system status
- A Wishbone bus in the design to set and monitor register values

Setting Serdes Debug Registers

You can adjust a large variety of serdes settings while running tests. Settings can be changed individually or by importing previously saved settings. You can also restore the design's original values from the configuration SRAM.

You can also monitor the following items in the Serdes Debug tab (green for good, red for bad):

- PLL lock status
- Receive input signal status
- CDR loss status

To set the serdes registers:

1. Click the **Serdes Debug** tab.
2. Click the DCUA drop-down menu and choose the block.
3. Click the Channel drop-down menu and choose the channel.
4. Change settings as desired.
5. Click **Apply**.

   The new values are immediately written to the FPGA.

To use previously saved settings:

1. Click the **Serdes Debug** tab.
2. Click the DCUA drop-down menu and choose the block.
3. Click the Channel drop-down menu and choose the channel.
4. Click **Import**.

5. In the Import SERDES File dialog box, browse to the desired serdes register (.srv) file.
6. Click **Open**.

   The file's settings are appear in the Serdes Debug tab.
7. Click **Apply**.

   The new values are immediately written to the FPGA.

To restore the design's original values:

1. Click the **Serdes Debug** tab.
2. Click the DCUA drop-down menu and choose the block.
3. Click the Channel drop-down menu and choose the channel.
4. Click **Config SRAM Reload**.

   The original settings appear in the Serdes Debug tab.
5. Click **Apply**.

   The new values are immediately written to the FPGA.

See Also ▶"Saving Serdes Debug Settings" on page 1100

Reading and Writing with a Wishbone Bus

If you have a Wishbone bus in your design, you can write directly into the serdes registers, manually or using a file, and read values from them.

To write to a serdes register:

1. Click the **Wishbone Debug** tab.
2. Enter the address of the register.
3. In the Data box, type a 1-byte value.
4. Click **Write**.

   The value in the Data box is written to the register.

To write to several serdes registers from a file:

1. Click the **Wishbone Debug** tab.
2. Click **Load**.

   The Load SERDES File dialog box opens.
3. Browse to the desired Tcl file.
4. Click **Open**.

   The addresses and values defined in the file are written to the FPGA.

To read a serdes register:

1. Click the **Wishbone Debug** tab.
2. Enter the address of the register.
3. If you want to monitor the value in the register continuously, select **Continuous**.
4. Click **Read**.

   The value in the register appears in the Data box. If you selected Continuous, the value changes as the register changes.

Saving the Reveal Analyzer Settings and Data

You can save the waveform data in different formats:

- Reveal Analyzer (.rva) file, which will also include the trigger settings and waveform setup for future use
- Value change dump (.vcd) file, which can be imported by third-party tools such as ModelSim or Active-HDL
- Text (.txt) file

Save to an .rva file if you want to see the data in the LA Waveform view again or if you want to use the trigger settings again. See “Saving a Reveal Analyzer File” on page 1099. To save to a different format, see “Saving to Other Formats” on page 1100.

Saving a Reveal Analyzer File

You can save the waveform along with the trigger settings and waveform setup in a Reveal Analyzer (.rva) file that you can use in the future. You can also save an existing .rva file in a file with a different name.

To save changes in the current file:

- Choose **File >**  **Save** <*file*>.

To save the file with a different name:

1. Choose **File > Save** <*file*> **As**.

   The Save Reveal Analyzer File dialog box appears.
2. Browse to the directory in which you want to save the project.
3. In the File name box, type the file name.
4. Click **Save**.

Saving to Other Formats

You can export the data captured for individual modules to a value change dump (.vcd) file, which can be imported by third-party tools such as ModelSim or Active-HDL, or to an ASCII text (.txt) file.

To export data:

1. Choose the module from the drop-down menu in the Reveal Analyzer tool bar.
2. If you want the data to include an approximate measure of time instead of a simple count of clock cycles, right-click the waveform and choose **Set Clock Period**. See “Specifying the Clock Period” on page 1101.
3. If you want to export only some of the signals, select them in the waveform. You can only export whole buses. If you select only some of the signals in a bus, you get the whole bus.
4. Right-click in the waveform and choose **Export Waveform**.

   The Export Waveform dialog box opens.
5. Browse to the location where you want to export the file.
6. Type in a name in the **File name** box.
7. Choose a file type.
8. If you are exporting only some of the signals, choose **Selected signals** in the Range box.
9. If you are exporting to .vcd, type in a module name. This will form the title in the .vcd file. If you leave the field empty, the module name will be “<unknown>”.
10. Click **Save**.

Saving Serdes Debug Settings

You can save your Serdes Debug settings for later use.

To save the Serdes Debug settings:

1. In the Serdes Debug tab, click **Export**.

   The Export SERDES Register File dialog box opens.
2. Browse to where you want to save the file.
3. Type in a name in the **File name** box.
4. Click **Save**.

Specifying the Clock Period

By default, Reveal Analyzer refers to time by a count of the sample clock cycles. You can convert this to an approximate measure of time, in nanoseconds or picoseconds, by specifying the length of the clock period. When the waveform is exported, this measure of time is shown instead of a simple count of cycles.

To set the clock period:

1. Right-click the waveform and choose **Set Clock Period**.
2. Specify the scale by choosing a unit for the period in the drop-down menu on the right side of the dialog box. If you are specifying a frequency in the megahertz range, choose **ns**. If you are specifying a frequency in the gigahertz range, choose **ps**.
3. Place the cursor in either the Period or Frequency text box and type in the desired value. The other text box fills in automatically.

   Only integers are allowed. If you try to specify a frequency that would require a non-integer period, the period is truncated to an integer and the frequency is automatically adjusted. For example, typing 150 in the Frequency text box gives you a period of 6 and a frequency of 166.
4. Click **OK**.

Chapter 14

Applying Engineering Change Orders

Engineering Change Orders, or ECOs, are requests to make changes to your design after it has been placed and routed. The changes are written into the Native Circuit Description (.ncd) database file without requiring you to repeat the entire design implementation process. After applying ECOs, you can continue to generate bitstream/JEDEC files for your design.

ECOs are mainly used to correct errors found in the hardware model during debugging, or to facilitate design specification changes when problems are introduced with the integration of other FPGAs or components in your PC board design.

The Diamond design flow supports interactive ECO editing with the ECO Editor, which is a tabular user interface similar to Spreadsheet View. The ECO Editor includes sheets for editing I/O preferences, PLL preferences, and memory initialization values. These features enable you to edit the Native Circuit Description (.ncd) database file created by the Place & Route Design process. You can also sort the sysIO or PLL preference list and save the sorting order. The ECO Editor provides a Change Log window that enables you to track the changes between the modified .ncd file and the post-PAR .ncd file.

Note

After you have edited your post-PAR NCD file, your functional simulation and timing simulation will no longer match.

Editing sysIO and PLL Preferences in ECO Editor

The ECO Editor provides separate sheets for editing sysIO and PLL preferences. After a design is placed and routed, you can open the ECO Editor to edit these preferences. The changes you make in the ECO Editor are written into the Native Circuit Description (.ncd) database file without requiring you to go through the entire design implementation process.

To edit sysIO and PLL preferences, click the sysIO Setting tab or the PLL Programming tab at the bottom of the ECO Editor window.

Editing sysIO Preferences

You can use the ECO Editor to edit sysIO preferences after the design has been placed and routed.

To edit sysIO preferences:

1. In Diamond, make sure that you have already run the Place & Route Design process, and then choose **Tools > ECO Editor** .
2. At the bottom of the ECO Editor, click the **sysIO Setting** tab.

   In the sysIO Setting window, default values are shown in blue, values from design sources are shown in green, and modified values are shown in black. You can customize the color and font settings by choosing **Tools > Options** from the Diamond menu bar.

   You can only edit the values of cells that have white backgrounds.
3. In the sysIO Setting window, double-click the preference that you want to modify. Select a value from the drop-down menu or type a new value.
4. When you have finished, choose **File > Save <file_name>.ncd** to save the changes to the current .ncd file.

   After you save the file, you can select the **NCD Change Log** tab to view a summary of the changes.

Editing PLL Preferences

You can use the ECO Editor to edit PLL preferences after the design has been placed and routed.

To edit PLL preferences:

1. In the Diamond main window, make sure you have already run the Place & Route Design process, and then choose **Tools > ECO Editor**.
2. At the bottom of the ECO Editor, click the **PLL Programming** tab.

   In the PLL Programming window, you can edit the values of cells that have white backgrounds.
3. Double-click the PLL preference cell that you want to modify. Select a value from the drop-down menu or type a new value.
4. When you have finished, choose **File > Save <file_name>.ncd** to save the changes to the current .ncd file.

   After you save the file, you can select the **NCD Change Log** tab to view a summary of the changes.

Sorting a Preference List

In the sysIO Setting or PLL Programming window, you can sort the preference list by one or more columns. The sorting order is automatically saved. When you close and re-open ECO Editor, you will see the same sort order displayed.

To sort a list by one column:

1. Select the **sysIO Setting** or **PLL Programming** tab at the bottom of the ECO Editor window to display the preference list you want to sort.
2. In the preference list, click the heading of the column you want to sort.

   An up arrow is displayed in the heading, indicating that the list has been sorted by the column in ascending order. Click the heading again to sort the list in descending order.

To sort a list by more than one column:

1. Click the **sysIO Setting** or **PLL Programming** tab at the bottom of the ECO Editor window to display the preference list you want to sort.
2. Choose **View > Sort**  .
3. In the Sort dialog box, select the first column to sort by from the “Sort By” drop-down menu. Click **Ascending** or **Descending** to control the order of the sorted list.
4. Use the “Then by” boxes to select additional columns, as well as the Ascending/Descending option for each column.

   The list will be sorted first by the column specified in the “Sort By” box and then by the additional columns in the “Then By” boxes in sequence.
5. Click **OK**.

Setting Memory Initialization Values in ECO Editor

Memory Initialization is a feature that allows you to change initial values to EBRs (Embedded Block RAM) and User Flash Memory (UFM) for device configuration after the design has been placed and routed. It saves you time by allowing you to fix a few incorrect initialization values in your post-PAR Native Circuit Description (.ncd) file without having to repeat the entire flow.

Note

User Flash Memory (UFM) is only available for MachXO2-640 and higher density devices.

Changing initial memory values might become necessary, for example, if incorrect values were assigned during synthesis or if there is some off-chip device that your FPGA is communicating with that requires a different value than the one that was initially set.

You use the ECO Editor's Memory Initialization window to update initial memory values.

See Also "Memory Initialization Input Files" on page 1105

- "Updating Initial Values for EBRs" on page 1105
- "Updating Initial Values for User Flash Memory" on page 1107

Memory Initialization Input Files

Setting initial memory values actually writes the initial values directly into your post-PAR Native Circuit Description (.ncd) file. For this to be accomplished, you need to provide the following information:

- Native Circuit Description file (.ncd) – the <*design_name*>.ncd file that is updated by the Place & Route Design process.
- Memory initialization file (.mem) – an ASCII-based text file that contains initial data values on various address locations in the memory module. This file should be in the same format that was used in module creation.

  Note

  You must create .mem files for ROMs. Memory Initialization files are optional for RAM modules.

  To learn more about how to create a memory initialization file, see "Creating a Memory Initialization File" on page 203
- Memory block parameters – information on the memory block such as module name, memory mode, depth, width, and format.

  If the memory module is created by IPexpress, the software can read the parameters from the Lattice parameter configuration file (.lpc). If your design utilizes PMI (Parameterized Module Instantiation) to generate memory modules, the software gets the parameters from the PMI module.
- Instance name (not a file) – the name that was given to the memory module in your top-level EDIF netlist.

Updating Initial Values for EBRs

Before updating memory values in ECO Editor, you need to add the memory module to your design and run the Place & Route Design process. The memory modules can be PMI modules or modules created by IPexpress.

For memory modules created by IPexpress, an LPC file is needed for memory initialization. PMI modules do not require an LPC file to re-initialize the memory.

Note

Once you edit initial memory values in the post-PAR NCD file, your functional simulation and timing simulation results will no longer match. You might need to preserve an original version of your NCD file.

To update memory initialization values:

1. Customize a memory initialization (.mem) file in any ASCII text editor or use the Memory Generator, the recommended method (see"Creating a Memory Initialization File" on page 203). In this step, you should change the values of the INITVALs in your .mem file as necessary and save the file in your project directory. You might also want to create a copy of the original .mem file.

   Note

   You must create a .mem file for ROM instances in your design. RAM instances may or may not need .mem files associated with them.

   For example, there might be a ROM memory instance in your design; so you create a .mem file called romb.mem in your project directory. These updated INITVALs will be used to override present ones that were auto-generated through the flow. Although the .mem file can be located anywhere on your system, it is recommended that you place them in your project directory.

2. In the Diamond main window, make sure that you have already successfully run the Place & Route Design process, and then choose **Tools > ECO Editor**.

3. Select the **Memory Initialization** tab at the bottom of the ECO Editor window.

   The Memory Initialization window displays the memory instances, memory file, and any LPC parameters for memory modules that were created by IPexpress.

4. Right-click a memory instance, and choose **Update Initial Memory** from the pop-up menu. You can also choose **Update Initial Memory** from the Edit menu.

   The Update Initial Memory dialog box opens, displaying the information about the currently selected memory block.

5. To change the format of the memory file, select a format from the File Format menu.

6. Click the **...** button next to the Memory File box to browse to the .mem file that you previously edited, as suggested in Step 1. Make sure that the File Format is correct.

   If your design targets a MachXO2 device and uses EBR blocks, you can use the .mem file or one of two other initialization options, as follows:

- Initialize to all 1s – fills the EBR memory with all 1s instead of the values in the .mem file.
- Initialize to all 0s – fills the EBR memory with all 0s instead of the values in the .mem file.
- Memory File – uses the values in the .mem file for initialization.

Note

The MacXO2 EBR cannot be initialized when the INIT_DATA attribute has been set to STATIC.

7. If the memory module is created by IPexpress, specify LPC parameters as follows. (This step is not required for PMI module.)
   - If the current design directory includes a .lpc file that is specified in the selected memory block, or if the parameters have been correctly written in the design source, the memory parameters from the .lpc file or the design source will be displayed in the dialog box.
   - If the associated .lpc file is stored outside the design directory, click the **...** button next to the LPC File box to browse to the file. The memory parameters included in the .lpc file will be displayed.
8. Click **Update**.

   The ECO Editor attempts to update the current memory block's initial memory values. A message will appear telling you whether the initialization succeeds or not.
9. Click **OK**.

   The Update Initial Memory dialog box closes after a successful initialization. If the initialization fails, the dialog box will stay open.
10. After the initialization succeeds, choose **File > Save File** to save the memory initialization changes to the current NCD file.

Updating Initial Values for User Flash Memory

The UFM (User Flash Memory) component is available in MachXO2-640 and higher density devices.

Before updating UFM initial values in ECO Editor, you need to use IPexpress to generate an EFB module with access to the user flash memory, add the module to your design, and run the Place & Route Design process.

To update UFM initialization values:

1. Customize a UFM memory initialization (.mem) file in any ASCII text editor or use the Memory Generator, the recommended method (see "Creating a Memory Initialization File" on page 203).

The UFM initialization file has the following properties and format:

Extension: .mem

Format: Binary, Hexadecimal

Data Width: 1 page (128 bits in one row of the file)

Number of Rows: less than or equal to the number of available pages

Example (Binary):

1010...1010 (Placed at the starting page of the UFM initialization data, address = N)
1010...1010 (Page address = N + 1)
1010...1010 (Page address = N + 2)
...
1010...1010 (Placed at the highest UFM page address)

Example (Hexadecimal):

A...B (Placed at the starting page of the UFM initialization data, address = N)
C...D (Page address = N + 1)
E...F (Page address = N +2)
...
A...F (Placed at the highest UFM page address)

The most significant byte (byte 15) of the page is on the left side of the row. The least significant byte of the page (byte 0) is on the right side of the row.

2. In the Diamond main window, make that sure you have already successfully run the Place & Route Design process, and then choose **Tools > ECO Editor** .

3. Select the **Memory Initialization** tab at the bottom of the ECO Editor window.

   The Memory Initialization window displays the UFM component in a tree view format. The window displays the UFM's number of pages and page range, as well as its initialization file name and format.

4. Right-click the UFM component, and choose **Edit UFM Parameters** from the pop-up menu. You can also choose **Edit UFM Parameters** from the Edit menu.

   The Update UFM Parameters dialog box opens, displaying the UFM information.

5. In the Update UFM Parameters dialog box, do one of the following:

   - If you want to initialize the UFM to all zeros, select **Enabled** for the Initialized with All Zeroes option.
   - If you want to change the UFM's initialization file, clear the Initialized with All Zeroes option, and then click the **...** button next to the UFM

Initialization Data File box to browse to the .mem file that you edited in Step 1. Make sure that the Initialization Data Format is correct.

Note

The default location where Lattice Diamond searches for the UFM initialization file is the project implementation directory. If the initialization file you specify is not in the implementation directory, ECO Editor will copy it to the implementation directory and inform you that the file is copied.

6. Click **Update**.

   The ECO Editor attempts to update the UFM's initial memory values. A message will appear telling you whether the initialization succeeds or not.

7. Click **OK**.

   The Edit UFM Parameters dialog box closes after a successful initialization. If the initialization fails, the dialog box will stay open.

8. After the initialization succeeds, choose **File > Save File** to save the memory initialization changes to the current NCD file.

Modifying SerDes Settings

For ECP5 devices, ECO Editor enables you to modify the SerDes settings for the PCS modules in your design after placement and routing.

To modify SerDes settings:

1. In ECO Editor, select the **SERDES Settings** tab.

   The SERDES page shows each PCS module in the design, with a tree view of the instance setup and each SerDes channel.

2. Double-click the name of the PCS module that you would like to modify to open the tree view.

   The tree view displays the attributes for the Instance Setup and for the Control and SerDes Setup for each channel.

3. Select the desired setting from the Attributes column for each attribute that you want to modify.

4. Save your changes.

See Also ▶"PCS Options (ECP5UM)" on page 1556

Adding Signal Probes

Lattice Diamond enables you to use the ecoeditor_probe attribute in your HDL code to mark signals as possible probe points. The attribute for each signal is then passed to the .ncd file when you process the design. You can then use the ECO Editor to assign these signals to external pins as test probes for

debugging. You do not need to open EPIC to configure the signal probes. The ECO Editor automatically gathers these signals from the .ncd file and adds them to the top of the list in the Signal Probes sheet.

Marking Signals as Possible Probe Points

In order to use the ECO Editor for configuring signal probes, you must edit the HDL source. Add the ecoeditor_probe attribute to each signal that you want to specify as a possible probe point, and add the syn_keep attribute to keep the specified net intact during optimization and synthesis.

To mark signals as possible probe points:

1. Open the HDL source and add the ecoeditor_probe attribute, using the appropriate VHDL or Verilog syntax, as shown in the following examples.

   Using VHDL:

```
attribute ecoeditor_probe : string;
attribute ecoeditor_probe of countai: signal is
"my_vhdl_signal";
attribute syn_keep : boolean;
attribute syn_keep of countai: signal is true;
```

   Using Verilog:

```
reg ack_err; /* synthesis syn_keep = 1 ecoeditor_probe =
"my_verilog_signal2" */;
```

2. Save the HDL file, and then process the design through Place & Route.

See Also ▶"Assigning Signals to Pins as Test Probes" on page 1110

Assigning Signals to Pins as Test Probes

The Signal Probes sheet in the ECO Editor lists all of the signal names. Those that were marked with the ecoeditor_probe attribute in the HDL are placed at the top of the list. An Available IO column is provided that enables you to assign pins to these signals and use them as test probes for debugging.

To assign signals to pins for use as test probes:

1. Select the Signal Probes tab.
2. For each signal that you want to assign, do the following:
   a. Double-click the cell in the Available IO column.

b. Select a pin for the signal from the drop-down list.

3. When you have finished making your selections, click the **Apply Signal Probes** button (**SP**) on the ECO Editor vertical toolbar.

   The ECO Editor routes the signals to the pins that you selected.

After applying the signal probes, you can view the routing in Physical View.

Tracking NCD Changes

All ECO changes that have been saved to the NCD file are recorded in the ECO Editor Change Log in the Reports view. Each time the NCD is modified and saved, a new change entry is added to the change log. All changes remain in the log even when the ECO Editor is closed.

To view the change log, select **ECO Editor Change Log** in the Tool Reports section of the Reports view.

Exporting sysIO Preferences

You can export the sysIO preferences currently displayed in the ECO Editor to a preference file (.lpf).

To export the current sysIO preferences:

1. In the ECO Editor, choose **File > Export > Preferences**.

   By default, the ECO Editor uses <project name>_eco to prevent the original .lpf file from being overwritten.

2. In the Save Preference File dialog box, accept the default file name or specify a different .lpf file name and location, and then click **Save**.

Running Design Rule Check

When you edit preferences in the ECO Editor, the software automatically runs design rule check and reports errors, if any, in the Diamond Output view.

You can also manually run the design rule check by choosing **View > Run DRC**  .

Chapter 15

Strategy Reference Guide

A strategy provides a unified view of all the options related to implementation tools such as synthesis, map, and place and route. Strategy options are listed in the Strategy dialog box. Open the dialog box by double-clicking a strategy name in the File List view.

For information about an option, select it. A brief description appears at the bottom of the dialog box. Press **F1** to open this guide and see the full description.

The descriptions of the strategy options are grouped by their processes (such as Synplify Pro, Translate Design, or Map Trace). Below is an alphabetical list of all the options.

For detailed information on how to apply strategies to your project, see “Using Strategies” on page 48.

A

- Address
- All Performance Grade
- Allow Duplicate Modules (for LSE)
- Allow Duplicate Modules (for Synplify Pro)
- Analysis Options (for Map Trace)
- Analysis Options (for Place & Route Trace)
- Area (for Precision)
- Area (for Synplify Pro)
- Arrange VHDL Files
- Array Bounds Case
- Auto Hold-Time Correction
- Auto Resource Allocation of RAM
- Auto Timing (for Map Trace)
- Auto Timing (for Place & Route Trace)

B

- Byte Wide Bit Mirror

C

- Carry Chain Length
- Chain Mode
- Check Unconstrained Connections (for Map Trace)
- Check Unconstrained Connections (for Place & Route Trace)
- Check Unconstrained Paths (for Map Trace)
- Check Unconstrained Paths (for Place & Route Trace)
- Clock Conversion
- Clock Skew Minimization
- Command Line Options (for LSE)
- Command Line Options (for Map Design)
- Command Line Options (for Place & Route Design)
- Command Line Options (for Precision)
- Command Line Options (for Synplify Pro)
- Command Line Options (for Synplify Pro Translate Design)
- Congestion-Driven Placement
- Congestion-Driven Routing
- Consistent Bus Name Conversion
- Convert Gated Clock
- Create Bit File
- Create Delay Statistic File

D

- Decode Unreachable States
- Default Clock Frequency
- Default Enum Encoding
- Disable Distributed RAM
- Disable I/O Insertion (for Precision)
- Disable Register Replication During S/R Optimization
- Disable IO Insertion (for Synplify Pro)
- Disable Timing Driven
- Disable UES
- DSP Style
- DSP Utilization

E

- EBR Utilization
- EDIF Names
- Enable I/O in Re-configuration
- Enable Timing Check
- Export Diamond Settings to Synplify Pro GUI
- External Clock

F

- Fanout Limit
- Fix Gated Clocks
- Force GSR (for LSE)
- Force GSR (for Precision)
- Force GSR (for Synplify Pro)
- Frequency (for Precision)
- Frequency (for Synplify Pro)
- FSM Encoding (for Precision)
- FSM Encoding (for Synplify Pro)
- FSM Encoding Style
- Full Case
- Full Name (for Map Trace)
- Full Name (for Place & Route Trace)

G

- Generate PUR in the Netlist
- Generate TRACE Report for Each Iteration
- Generate X for Setup/Hold Violation
- Grant Timeout

H

- Hardware Evaluation (for LSE)
- Hardware Evaluation (for Translate Design)
- Hold Check Min Speed Grade

I

- Ignore Preference Errors (for Map Design)
- Ignore Preference Errors (for Place & Route Design)
- Ignore RAM Read/Write Collision
- Infer DSPs Across Hierarchy
- Infer GSR
- Input Delay (ns)
- Intermediate File Dump
- IO Registering

L

- LengthBits
- Library Directories
- Loop Limit

M

- Macro Search Path (for LSE)
- Macro Search Path (for Translate Design)
- Max Fanout Limit
- Memory Initial Value File Search Path (for LSE)
- Memory Initial Value File Search Path (for Translate Design)
- Multichip Module Prefix
- Multi-Tasking Node List
- MUX Style

N

- Negative Setup-Hold Times
- No Header
- Number of Critical Paths (for LSE)
- Number of Critical Paths (for Precision)
- Number of Critical Paths (for Synplify Pro)
- Number of Start/End Points
- Number of Timing Summary Paths
- Number of Unconstrained Paths (for Map Trace)
- Number of Unconstrained Paths (for Place & Route Trace)

O

- Optimization Goal
- Output Delay (ns)
- Output Format
- Output Netlist Format (for Precision)
- Output Netlist Format (for Synplify Pro)
- Output Preference File
- Output Zero Frames
- Overmap Device if Design Does Not Fit

P

- Pack Logic Block Util.
- Parallel Case
- Partition Search Path
- Path-based Placement
- Pipelining and Retiming
- Placement Effort Level
- Placement Iteration Start Pt
- Placement Iterations
- Placement Save Best Run
- Placement Sort Best Run
- PROM Data Output Format
- Propagate Constants
- Push Tristates

R

- RAM Style
- ReadBack
- ReadCapture
- Reencode FSM Outputs
- Register Configuration
- Register Retiming
- Remove Duplicate Registers
- Remove LOC Properties (for LSE)
- Remove LOC Properties (for Translate Design)
- Remove Previous Design Directory
- Report Asynchronous Timing Loops (for Map Trace)
- Report Asynchronous Timing Loops (for Place & Route Trace)
- Report Clock Frequencies
- Report Missing Constraints
- Report Signal Cross Reference
- Report Style (for Map Trace)
- Report Style (for Place & Route Trace)
- Report Symbol Cross Reference
- Report Timing Violations
- Reset Config RAM in Re-configuration
- Resolved Mixed Drivers (for Synplify Pro)
- Resolved Mixed Drivers (for LSE)
- Resource Sharing (for LSE)
- Resource Sharing (for Precision)
- Resource Sharing (for Synplify Pro)
- Retarget Speed Grade
- ROM Style
- Route Estimation Algorithm (%)
- Routing Delay Reduction Passes
- Routing Method
- Routing Passes
- Routing Resource Optimization
- Run DRC
- Run Placement Only
- Run Time Reduction
- Run Retiming

S

- Search Path
- Show Clock Domain Crossing
- Show Net Fanout
- SLVS MIPI LVCMOS Output
- Speed for Hold Analysis
- Speed for Setup Analysis
- SPI Start Address
- Startup Clock
- Stop Once Timing is Met
- Sysbus Clock Config
- Sysbus Config

T

- Target Frequency
- Timing Driven Mapping
- Timing Driven Node Replication
- Timing Driven Packing
- Timing Simulation Max Delay between Buffers (ps)
- Transform Set/Reset on DFFs to Latches
- Transport Mode of Path Delay in VHDL

U

- UFM Initval Order
- Update Compile Point Timing Data
- Use Carry Chain
- Use Clock Period for Unconstrained I/O
- Use IO Insertion
- Use IO Registers
- Use LPF Created from SDC in Project (for LSE)
- Use LPF Created from SDC in Project (for Synplify Pro)
- Use Safe FSM

V

- Verilog Hierarchy Separator
- Verilog Input
- Verilog Standard V2001
- VHDL 2008 (for LSE)
- VHDL 2008 (for Synplify Pro)

W

- Wait State Timeout
- Worst-Case Paths (0-4096) (for Map Trace)
- Worst-Case Paths (0-4096) (for Place & Route Trace)
- Write Verbose Netlist

X

- XRES Calibration Control

Synplify Pro Options

This page lists all the strategy options associated with the Synplify Pro Synthesis process.

For information on their use in Synplify Pro, see the Synopsis Synplify Pro for Lattice Reference Manual.

- Allow Duplicate Modules (for Synplify Pro)
- Area
- Arrange VHDL Files
- Clock Conversion
- Command Line Options

- Default Enum Encoding
- Disable IO Insertion
- Disable Register Replication During S/R Optimization
- EDIF Names
- Export Diamond Settings to Synplify Pro GUI
- Fanout Limit
- Force GSR
- Frequency
- FSM Encoding
- Library Directories
- Number of Critical Paths
- Number of Start/End Points
- Output Netlist Format
- Pipelining and Retiming
- Push Tristates
- Resolved Mixed Drivers (for Synplify Pro)
- Resource Sharing
- Update Compile Point Timing Data
- Use Clock Period for Unconstrained I/O
- Use LPF Created from SDC in Project (for Synplify Pro)
- Verilog Input
- VHDL 2008 (for Synplify Pro)

Allow Duplicate Modules (for Synplify Pro) Allows the use of duplicate modules in your design.

When it is set to True, the last definition of the module is used by the software and any previous definitions are ignored. The default is False.

Area (for Synplify Pro) Specifies optimization preference for area reduction over timing delay reduction.

The True option specifies the area reduction mode. When set to True, this setting overrides the setting in Frequency (for Synplify Pro).

The default is False for all devices.

This option is equivalent to the “set_option -frequency 1” command in Synplify Pro.

Arrange VHDL Files Allows Synplify Pro to reorder the VHDL source files for synthesis.

The default is True for VHDL or Schematic/VHDL design entry type projects, and False for other projects. When this is set to False, Synplify Pro will use the file order in the Diamond File List view.

Clock Conversion Controls gated and generated clock conversion.

Values are:

- 0 – does not convert
- 1 (default) – converts; does not report
- 2 – converts; reports only sequential elements that could not be converted
- 3 – converts; reports all sequential elements

This option is equivalent to the “set_option -fix_gated_and_generated_clocks 0 | 1 | 2 | 3” command in Synplify Pro.

Command Line Options (for Synplify Pro) Enables additional command line options for the Synplify Pro Synthesis process.

To enter a command line option:

1. In the Strategy dialog box, select **Synplify Pro** in the Process list.
2. Double-click the Value column for the Command Line Options option.
3. Type in the option and its value (if any) in the text box.
4. Click **Apply**.

For example:

```
set_option -library_path c:/source
```

Default Enum Encoding (For VHDL designs) Defines how enumerated data types are implemented.

The type of implementation affects the performance and device utilization. Available options are:

- Default – Automatically assigns an encoding style based on the number of states:
  - Sequential: 0-4 enumerated types
  - Onehot: 5-40 enumerated types
  - Gray: more than 40 enumerated types
- Gray – Only one bit of the state register changes at a time, but because more than one bit can be hot, the value must be decoded to determine the state. For example: 000, 001, 011, 010, 110
- Onehot – Only two bits of the state register change (one goes to 0; one goes to 1) and only one of the state registers is hot (driven by a 1) at a time. For example: 0000, 0001, 0010, 0100, 1000

- Sequential – More than one bit of the state register can change at a time, but because more than one bit can be hot, the value must be decoded to determine the state. For example: 000, 001, 010, 011, 100

This option is equivalent to the “set_option -default_enum_encoding default | onehot | gray | sequential” command in Synplify Pro.

Disable IO Insertion (for Synplify Pro) Controls whether the synthesis tool will add I/O buffers into your design.

If this is set to True, Synplify Pro will not add I/O buffers into your design. If it is set to False (default), the synthesis tool will insert I/O buffers into your design.

This option is equivalent to the “set_option -disable_io_insertion 1 | 0” command in Synplify Pro.

Disable Register Replication During S/R Optimization When this option is set in the strategy, the synthesis tool will NOT duplicate the registers while inferring the address pointers for the RAM during Shift-register inference. By default this option is OFF and the tool will duplicate the address pointer registers to get better performance.

EDIF Names Controls the format of bus names in the EDIF file. Select Standard to use brackets in the names of bus signals as in "bus[1]", which is the current standard for Synplify Pro. Select Legacy to use the older format as in "bus_1".

Standard is required to use Synplify Pro to cross-probe from the Place & Route TRACE Report to a schematic view. See “Cross-Probing to Synplify Pro HDL Analyst” on page 718.

Export Diamond Settings to Synplify Pro GUI Controls whether the strategy settings are exported to Synplify Pro during interactive synthesis (opening Synplify Pro through the Tools menu). After opening Synplify Pro, you can change settings in Synplify Pro's interface. This option has no effect with integrated or stand-alone synthesis.

Available options are:

- No (default) – Synplify Pro opens with its own defaults, ignoring the strategy settings.
- Yes – Synplify Pro opens with the strategy settings every time. Options set and saved in a previous Synplify Pro session are ignored.
- Only on First Launch – Synplify Pro opens with the strategy settings the first time only. After that, Synplify Pro opens with settings saved in a previous session or with its own defaults. After the first time, the strategy settings are ignored.

For more information, see “Interactive Synthesis” on page 585.

Fanout Limit Controls fanout during synthesis. When the specified fanout limit is achieved, logic will be duplicated.

The default is 1000.

This option is equivalent to the “set_option -maxfan <*number*>” command in Synplify Pro.

Force GSR (for Synplify Pro) Forces Global Set/Reset Pin usage.

Available options are:

- Auto – Allows the software to decide whether to infer Global Set/Reset in your design.
- False (default) – Does not infer Global Set/Reset in your design.
- True – Always infers Global Set/Reset in your design.

This option is equivalent to the “set_option -force_gsr auto | yes | no” command in Synplify Pro.

Frequency (for Synplify Pro) Specifies the global design frequency (in MHz). Nothing in the Value column means "auto" (the default) and Synplify Pro will try to maximize the frequency of the clocks.

The setting is ignored when Area (for Synplify Pro) is set to True.

This option is equivalent to the “set_option -frequency <*number*> | auto” command in Synplify Pro.

FSM Encoding (for Synplify Pro) Enables or disables the FSM Compiler and controls the use of FSM synthesis for state machines.

When Synplify Pro is selected as the synthesis tool, it enables or disables the FSM Compiler and controls the use of FSM synthesis for state machines. When this is set to True (default), the FSM Compiler automatically recognizes and optimizes state machines in the design. The FSM Compiler extracts the state machines as symbolic graphs, and then optimizes them by re-encoding the state representations and generating a better logic optimization starting point for the state machines.

This option is equivalent to the “set_option -symbolic_fsm_compiler 1 | 0” command in Synplify Pro.

Library Directories Specifies all the paths to the directories which contain the Verilog library files to be included in your design for the project.

You can also add custom library files with module definitions for the design in a single file. The names of files read from the library path must match module names. Mismatches result in error messages.

Number of Critical Paths (for Synplify Pro) Specifies the number of critical timing paths to be reported in the timing report.

This option is equivalent to the “set_option -num_critical_paths <*number*>” command in Synplify Pro.

Number of Start/End Points Specifies the number of start and end points you want the software to report in the critical path section of the timing report.

This option is equivalent to the “set_option -num_startend_points <*number*>” command in Synplify Pro.

Output Netlist Format (for Synplify Pro) Outputs a mapped Verilog or VHDL netlist for post-synthesis simulation.

Available options are: None (default), VHDL, and Verilog.

This option is equivalent to the “set_option -write_verilog 1 | 0 -write_vhdl 1 | 0” command in Synplify Pro.

Pipelining and Retiming Enables the pipelining and retiming features to improve design performance.

Values are:

- None – Disables the pipelining and retiming features.
- Pipelining Only (default) – Runs the design at a faster frequency by moving registers into the multiplier, creating pipeline stages.
- Pipelining and Retiming – When enabled, registers may be moved into combinational logic to improve performance.

This option is equivalent to the “setup_option -pipe 1 | 0 -retiming 1 | 0” command in Synplify Pro.

Push Tristates When this is set to True, the Synplify Pro compiler pushes tristates through objects such as muxes, registers, latches, buffers, nets, and tristate buffers, and propagates the high impedance state.

The high-impedance states are not pushed through combinational gates such as ANDs or ORs.

The default is False.

This option is equivalent to the “set_option -compiler_compatible 1 | 0” command in Synplify Pro.

Resolved Mixed Drivers (for Synplify Pro) If a net is driven by a VCC or GND and active drivers, setting this option to True (default) will connect the net to the VCC or GND driver.

This option is equivalent to the “set_option -resolve_multiple_driver 1 | 0” command in Synplify Pro.

Resource Sharing (for Synplify Pro) When this is set to True (default), the synthesis tool uses resource sharing techniques to optimize area.

With resource sharing, synthesis uses the same arithmetic operators for mutually exclusive statements; for example, with the branches of a case

statement. Conversely, you can improve timing by disabling resource sharing, but at the expense of increased area.

This option is equivalent to the “set_option -resource_sharing 1 | 0” command in Synplify Pro.

Update Compile Point Timing Data Determines whether (True) or not (False) changes inside a compile point can cause the compile point (or top-level) containing it to change accordingly.

When this is set to False (default), Synplify Pro keeps the top level module the same, which is desired by incremental flow.

When this is set to True, changes in low level partitions will be propagated to top partitions up to top module. Synplify Pro will possibly optimize timing data and certainly will write a new timestamp onto the partition for the top level module.

This option is equivalent to the “set_option -update_models_cp 1 | 0” command in Synplify Pro.

Use Clock Period for Unconstrained I/O Controls whether to forward annotate constraints for I/O ports without explicit user-defined constraints.

When this is set to True, only explicit I/O port constraints are forward annotated. When it is set to False (the default), all I/O port constraints are forward annotated.

This option is equivalent to the “set_option -auto_constraint_io 1 | 0” command in Synplify Pro.

Use LPF Created from SDC in Project (for Synplify Pro) When this is set to True (default), Synplify Pro creates a preference (.lpf) file based on the Synopsys Design Constraint (.sdc) file. With this .lpf file, the synthesis constraints will also be applied to the Map Design stage of implementation.

This option is equivalent to the “set_option -write_apr_constraint 1 | 0” command in Synplify Pro.

Verilog Input Specifies the Verilog standard used for the project.

The default is System Verilog.

This option is equivalent to the “set_option -vlog_std v2001 | v95 | sysv” command in Synplify Pro.

For information about Verilog 2001, refer to the Synplify and Synplify Pro for Lattice Reference Manual. Go to the “Verilog 2001 Support” section in the “Verilog Language Support” chapter.

VHDL 2008 (for Synplify Pro) When this is set to True, VHDL 2008 is selected as the VHDL standard for the project.

Precision Options

This page lists all the strategy options associated with the Precision Synthesis process.

- Area
- Array Bounds Case
- Auto Resource Allocation of RAM
- Command Line Options
- Convert Gated Clock
- Disable I/O Insertion
- Force GSR
- Frequency
- FSM Encoding
- Full Case
- Ignore RAM Read/Write Collision
- Infer DSPs Across Hierarchy
- Input Delay (ns)
- Number of Critical Paths
- Number of Timing Summary Paths
- Output Delay (ns)
- Output Netlist Format
- Output Preference File
- Parallel Case
- Reencode FSM Outputs
- Report Clock Frequencies
- Report Missing Constraints
- Report Timing Violations
- Resource Sharing
- Run Retiming
- Show Clock Domain Crossing
- Show Net Fanout
- Transform Set/Reset on DFFs to Latches
- Use Safe FSM
- Verilog Standard V2001

Area (for Precision) Specifies optimization preference for area reduction over timing delay reduction.

The True option specifies the area reduction mode. When set to True, this option overrides the setting in Frequency (for Precision).

The default is True for MachXO, and False for the other devices.

This option is equivalent to the “setup_design -compile_for_area” command in Precision RTL Synthesis.

Array Bounds Case Enables or disables the Precision tool to check array bounds for Verilog input files.

The default is False.

This option is equivalent to the “setup_design –array_bounds_check” command in Precision RTL Synthesis.

Auto Resource Allocation of RAM Enables or disables the automatic resource allocation of RAMs.

When set to False, it stops the synthesis step when the number of inferred RAMs exceeds the block-RAM resources on the target device. To map RAMs to block-RAM resources first and then map remaining RAM structures into logic resources, set this to True.

This option is equivalent to the “setup_design -auto_resource_allocation_ram” command in Precision RTL Synthesis.

Command Line Options (for Precision) Enables additional command line options for the Precision Synthesis process.

To enter a command line option:

1. In the Strategy dialog box, select **Precision** in the Process list.
2. Double-click the Value column for the Command Line Options option.
3. Type in the option and its value (if any) in the text box.
4. Click **Apply**.

Convert Gated Clock Controls the gated-clock conversion by Precision RTL Synthesis during the synthesis phase.

Options are:

- True – Converts gated-clock flip-flops to equivalent enable flip-flops.
- False (default) – Does not convert gated-clock flip-flops to equivalent enable flip-flops.

This option is equivalent to the “setup_design -gated_clock” command in Precision RTL Synthesis.

Disable I/O Insertion (for Precision) Controls whether the synthesis tool will add I/O buffers into your design.

If this is set to True, Precision RTL Synthesis will not add I/O buffers into your design. If it is set to False (default), the synthesis tool will insert I/O buffers into your design.

This option is equivalent to the “setup_design -addio” command in Precision RTL Synthesis.

Force GSR (for Precision) Forces Global Set/Reset Pin usage.

Available options are:

- Auto – Allows the software to decide whether to infer Global Set/Reset in your design.
- True – Always infers Global Set/Reset in your design.
- False (default) – Does not infer Global Set/Reset in your design.

This option is equivalent to the “setup_design -infer_gsr” command in Precision RTL Synthesis.

Frequency (for Precision) Specifies the global design frequency (in MHz).

The default is 200 for both Precision RTL Synthesis and Synplify Pro. The option is ignored when Area (for Precision) is set to True.

This option is equivalent to the “setup_design -frequency=<*number*>” command in Precision RTL Synthesis.

FSM Encoding (for Precision) Specifies the encoding style to use with the design.

When Precision RTL Synthesis is selected as the synthesis tool, it specifies the encoding style to use with the design. Valid values are: Auto (default), Binary, One Hot, Two Hot, Random, and Gray.

This option is equivalent to the “setup_design -encoding=auto | binary | onehot | twohot | random | gray” command in Precision RTL Synthesis.

Full Case When set to True, tells Precision that for all case statements throughout your design, all relevant conditions are specified. If the default assignment is not used then this option prevents the inference of latches.

The default is False.

This option is equivalent to the “setup_design –variable_full_case=TRUE” command in Precision RTL Synthesis.

Ignore RAM Read/Write Collision Controls whether to ignore RAM read/write collision.

If the dual-port RAMs in your design do not require simultaneous read and write of the same address, you can set this to True to prevent inference of glue-logic and save logic resources. The default is False.

This option is equivalent to the “setup_design -ignore_ram_rw_collision” command in Precision RTL Synthesis.

Infer DSPs Across Hierarchy When this is set to True, Precision Synthesis will optimize pipeline registers across hierarchical boundaries.

The True option performs one or more of the following actions:

- Absorb all necessary pipe registers
- Rearrange arithmetic operators where necessary
- Other optimization techniques

The default is False.

This option is equivalent to the “setup_design -dsp_across_hier” command in Precision RTL Synthesis.

Input Delay (ns) Specifies the global input delay (in ns) for use with the frequency switch.

This option is equivalent to the “setup_design -input_delay=<*number*>” command in Precision RTL Synthesis.

Number of Critical Paths (for Precision) Specifies the number of critical timing paths to be reported in the timing report.

The default is 3.

This option is equivalent to the “setup_analysis -critical_paths -num_critical_paths=<*number*>” command in Precision RTL Synthesis.

Number of Timing Summary Paths Specifies the number of timing paths to be reported in the timing report.

The default is 10.

This option is equivalent to the “setup_analysis -summary -num_summary_paths=<*number*>” command in Precision RTL Synthesis.

Output Delay (ns) Specifies the global output delay (in ns) for use with the frequency switch.

This option is equivalent to the “setup_design -output_delay=<*number*>” command in Precision RTL Synthesis.

Output Netlist Format (for Precision) Outputs a mapped Verilog or VHDL netlist for post-synthesis simulation.

Available options are: None (default), Verilog, and VHDL.

For Precision RTL Synthesis, setting this to Verilog will create a .vm file. Setting it to VHDL creates a .vhm file.

This option is equivalent to the “setup_design [-verilog | -vhdl]” command in Precision RTL Synthesis.

Output Preference File When this is set to True (default), Precision RTL Synthesis creates a preference file and gives it the same name as the module: module_name.prf.

This option is equivalent to the “setup_design -vendor_constraint_file=false | true” command in Precision RTL Synthesis.

Parallel Case When set to True, tells Precision that for all case statements throughout your design, case conditions are mutually exclusive. This results in a multiplexer implementation instead of a priority encoder.

The default is False.

This option is equivalent to the “setup_design –variable_parallel_case=TRUE” command in Precision RTL Synthesis.

Reencode FSM Outputs Controls Precision FSM extraction and re-encoding when Precision needs to create decoding logic.

The default state is True. To stop Precision from extracting, re-encoding, and creating decode logic, set this to False.

This option is equivalent to the “setup_design -reencode_fsm_outputs” command in Precision RTL Synthesis.

Report Clock Frequencies Setting this to True (default) lets the software report all clock frequencies.

This option is equivalent to the “setup_analysis -clock_frequency” command in Precision RTL Synthesis.

Report Missing Constraints When this is set to True, the timing report will report missing constraints.

The default is False.

This option is equivalent to the “setup_analysis -missing_constraints” command in Precision RTL Synthesis.

Report Timing Violations When this is set to True (default), the timing report will include timing violations.

This option is equivalent to the “setup_analysis -timing_violations” command in Precision RTL Synthesis.

Resource Sharing (for Precision) When this is set to True (default), the synthesis tool uses resource sharing techniques to optimize area.

With resource sharing, synthesis uses the same arithmetic operators for mutually exclusive statements; for example, with the branches of a case statement. Conversely, you can improve timing by disabling resource sharing, but at the expense of increased area.

This option is equivalent to the “setup_design -resource_sharing” command in Precision RTL Synthesis.

Run Retiming Setting this to True (default) causes the advanced retiming algorithms to be run.

This option is equivalent to the “setup_design -retiming” command in Precision RTL Synthesis.

Show Clock Domain Crossing When this is set to True, the timing report will show clock domain crossings.

The default is False.

This option is equivalent to the “setup_analysis -clock_domain_crossing” command in Precision RTL Synthesis.

Show Net Fanout When this is set to True (default), the timing report will show net fanout.

This option is equivalent to the “setup_analysis -net_fanout” command in Precision RTL Synthesis.

Transform Set/Reset on DFFs to Latches Setting this to True transforms Set/Reset on DFFs to Latches.

The default is True.

This option is equivalent to the “setup_design -transformations” command in Precision RTL Synthesis.

Use Safe FSM Enables or disables the use of safe FSMs.

Safe FSMs are FSMs that have no illegal states. The default is False.

This option is equivalent to the “setup_design -use_safe_fsm” command in Precision RTL Synthesis.

Verilog Standard V2001 When this is set to True, Verilog 2001 is selected as the Verilog standard for the project. Otherwise, Verilog 95 will be used.

This option is equivalent to the “setup_design -language_syntax_verilog” command in Precision RTL Synthesis.

LSE Options

This page lists all the strategy options associated with the Lattice Synthesis Engine (LSE) synthesis process.

- Allow Duplicate Modules (for LSE)
- Carry Chain Length
- Command Line Options

- Decode Unreachable States
- Disable Distributed RAM
- DSP Style
- DSP Utilization
- EBR Utilization
- Fix Gated Clocks
- Force GSR
- FSM Encoding Style
- Hardware Evaluation
- Intermediate File Dump
- Loop Limit
- Macro Search Path
- Max Fanout Limit
- Memory Initial Value File Search Path
- MUX Style
- Number of Critical Paths
- Optimization Goal
- Propagate Constants
- RAM Style
- Remove Duplicate Registers
- Remove LOC Properties
- Resolved Mixed Drivers (for LSE)
- Resource Sharing
- ROM Style
- Target Frequency
- Use Carry Chain
- Use IO Insertion
- Use IO Registers
- Use LPF Created from SDC in Project (for LSE)
- VHDL 2008 (for LSE)

Allow Duplicate Modules (for LSE)

When set to True, allows the design to keep duplicate modules. LSE issues a warning and uses the last definition of the module. Any previous definitions are ignored. The default is False, which causes an error if there are duplicate modules.

This option is equivalent to the “-allow_duplicate_modules” option in the SYNTHESIS command.

Carry Chain Length Specifies the maximum number of carry chain cells (CCUs) that get mapped to a single carry chain. Default is 0, which is interpreted as infinite length.

This option is equivalent to the "-carry_chain_length" option in the SYNTHESIS command.

Command Line Options (for LSE) Enables additional command line options for the LSE Synthesis process.

To enter a command line option:

1. In the Strategy dialog box, select **LSE** in the Process list.
2. Double-click the Value column for the Command Line Options option.
3. Type in the option and its value (if any) in the text box.
4. Click **Apply**.

For detailed descriptions of LSE command line options, see "Running SYNTHESIS from the Command Line" on page 2480.

Decode Unreachable States When set to True, synthesis infers safe recovery logic from unreachable states in all the state machines of the design.

This option is equivalent to the "-decode_unreachable_states" option in the SYNTHESIS command.

Disable Distributed RAM When set to True, inferred memory will not use the distributed RAM of the PFUs.

DSP Style Specifies how DSP modules should be implemented: with DSP resources or with Logic (LUTs).

This option is equivalent to the "-use_dsp" option in the SYNTHESIS command.

DSP Utilization Specifies the percentage of DSP sites that LSE should try to use.

This option is equivalent to the "-dsp_utilization" option in the SYNTHESIS command.

EBR Utilization Specifies EBR utilization target setting in percent of total vacant sites. LSE will honor the setting and do the resource computation accordingly. Default is 100 (in percentage).

This option is equivalent to the "-bram_utilization" option in the SYNTHESIS command.

Fix Gated Clocks When set to True, LSE changes standard gated clocks to forms more effective for FPGAs. Clocks are gated with AND or OR gates to conserve power, but in FPGAs such clocks cause skew and prevent global clock resources from being used. The Fix Gated Clocks option is ignored if the Optimization Goal option is set to Area.

The gated clocks must be specified in the .ldc file with create_clock constraints. For help writing the constraints, see "Defining Clocks Using LDC Editor" on page 369 or "create_clock" on page 1341. All inputs of the gating logic must be driven by primary inputs and the gating logic must be decomposable. Instantiated primitives and black boxes are not affected.

Converted clocks and the associated registers are reported in the synthesis.log file.

Force GSR (for LSE) Enables (True) or disables (False) forced use of the global set/reset routing resources. When the value is Auto, the synthesis tool decides whether to use the global set/reset resources.

This option is equivalent to the "-force_gsr" option in the SYNTHESIS command.

FSM Encoding Style Specifies the encoding style to use with the design.

This option is equivalent to the "-fsm_encoding_style" option in the SYNTHESIS command. Valid options are auto, one-hot, gray, and binary. The default value is auto, meaning that the tool looks for the best iimplementation.

Note

The encoding type "gray" only works with less than or equal to four machine states. When the number of machine states is large than four, LSE will use other encoding styles and issue the following warning message:

WARNING - Gray encoding is not supported for state machines with more than four states.

Hardware Evaluation (for LSE) Enables or disables the ability to temporarily test IP in a device without an IP license. If enabled, a timer is added to the design that allows unlicensed IP to function for about 4 hours in a device. If disabled, you cannot generate a bitstream if there are any unlicensed IP in the design.

You might want to disable this option to refine your design while waiting for the license. You will not be able to generate a bitstream, but you will be able to see how resources are used (without the timer) and close timing. When you get the license, you can then generate the bitstream.

Regardless of how this option is set, if there are any unlicensed IP in the design, some features of Diamond, such as gate level simulation and EPIC, are blocked.

This option is equivalent to the "-dt" option in the SYNTHESIS command.

Intermediate File Dump If you set this to True, LSE will produce intermediate encrypted Verilog files. If you supply Lattice with these files, they can be decrypted and analyzed for problems. This option is good for analyzing simulation issues.

This option is equivalent to the “-ifd” option in the SYNTHESIS command.

Loop Limit Specifies the maximum number of iterations of "for" and "while" loops in the source code. The limit is applied when the loop index is a variable, not when it is a constant. The higher the loop_limit, the longer the run time. The default value is 1950. Setting a higher value may cause stack overflow during some of the optimizations during synthesis. A lower value will be ignored and the default used instead.

This option is equivalent to the “-loop_limit” option in the SYNTHESIS command.

Macro Search Path (for LSE) Allows you to specify a path (or paths) to locate physical macro files used in a given design. The software will add the specified paths to the list of directories to search when resolving file references. The option can also be used for indicating the directories containing include files that are specified in the RTL design files.

You don’t need to specify a search path if the necessary .ngo or .nmc file is in the directory containing the top-level .ngo file or if the FILE attribute in the design gives a complete path name for the file (instead of a relative path name).

The software follows the following order to search for .ngo files:

1. Current implementation directory
2. Project directory
3. Directories where the LPC or IPX source files reside
4. User-specified macro search paths

To specify a macro search path, double-click the Value box, and directly enter the path or click the **...** button to browse for one or more paths.

This option is equivalent to the “-p” option in the SYNTHESIS command.

Max Fanout Limit Specifies the maximum fanout setting. LSE will make sure that any net in the design is not exceeding this limit. Default is 1000 fanouts.

This option is equivalent to the “-max_fanout” option in the SYNTHESIS command.

Memory Initial Value File Search Path (for LSE) Allows you to specify a path (or paths) to locate memory initialization file (.mem) used in a given design. The software will add the specified path(s) to the list of directories to search when resolving file references.

To specify a search path, double-click the Value box, and directly enter the path or click the **...** button to browse for one or more paths.

This option is equivalent to the “-p” option in the SYNTHESIS command.

MUX Style Specifies the MUX style setting, which controls the way the macrogenerator implements the multiplexer macros.

Valid options are:

- Auto (default) - LSE looks for the best implementation for each considered macro.
- L6Mux Multiple - Generates multiplexers allowing for multiple L6Mux resources.
- L6Mux Single - Generates multiplexers allowing for the use of a single L6Mux resource.
- PFU Mux - Generates multiplexers using only PFUMux and LUT4 resources.

Note

L6Mux resources will only be inferred when driven by four LUT4 and two PFUMux devices.

This option is equivalent to the “-mux_style” option in the SYNTHESIS command.

Number of Critical Paths (for LSE) Specifies the number of critical timing paths to be reported in the timing report.

This option is equivalent to the “-twr_paths” option in the SYNTHESIS command.

Optimization Goal Enables LSE to optimize the design for area, speed, or balanced.

Valid options are:

- Area – Optimizes the design for area by reducing the total amount of logic used for design implementation.

  When Optimization Goal is set to Area, LSE honors the LDC constraints if there are any. If Use IO Registers is set to Auto, LSE packs input and output registers into I/O pad cells.

  Note

  With the Area setting, LSE also ignores all SDC constraints. These constraints are not used by LSE and are not added to an .lpf file for use by the later stages of implementation.

- Timing – Optimizes the design for speed by reducing the levels of logic.

When Optimization Goal is set to Timing and a create_clock constraint is available in an .ldc file, LSE ignores the Target Frequency setting and uses the value from the create_clock constraint instead.

If there are multiple clocks, and if not all the clocks use create_clock constraint, then LSE will asssign 200 MHz constraint on the remaining clocks in Timing Mode.

If Use IO Registers is set to Auto, LSE does not pack input and output registers into I/O pad cells.

- Balanced – Optimizes the design for both area and timing.

  When Optimization Goal is set to Balanced, all timing driven optimizations based on static timing analysis will run depending on LDC constraints. If Use IO Registers is set to Auto, LSE does not pack input and output registers into I/O pad cells.

The default setting depends on the device type. Smaller devices, such as MachXO and Platform Manager, default to Balanced. Larger devices—ECP5U, LatticeECP2, LatticeECP3, and LatticeXP2—default to Timing.

For more information, see "Optimizing LSE for Area and Speed" on page 581.

This option is equivalent to the "-optimization_goal" option in the SYNTHESIS command.

Propagate Constants When set to True (default), enables constant propagation to reduce area, where possible. LSE will then eliminate the logic used when constant inputs to logic cause their outputs to be constant.

You can turn off the operation by setting this option to False.

This option is equivalent to the "-propagate_constants" option in the SYNTHESIS command.

RAM Style Sets the type of random access memory globally to distributed, embedded block RAM, or registers.

The default is Auto which attempts to determine the best implementation, that is, the synthesis tool will map to technology RAM resources (EBR/Distributed) based on the resource availability.

This option will apply a syn_ramstyle attribute globally in the source to a module or to a RAM instance. To turn off RAM inference, set its value to Registers.

- Registers – Causes an inferred RAM to be mapped to registers (flip-flops and logic) rather than the technology-specific RAM resources.
- Distributed – Causes the RAM to be implemented using the distributed RAM or PFU resources.
- Block_RAM – Causes the RAM to be implemented using the dedicated RAM resources. If your RAM resources are limited, for whatever reason, you can map additional RAMs to registers instead of the dedicated or distributed RAM resources using this attribute.

This option is equivalent to the “-ramstyle” option in the SYNTHESIS command.

Remove Duplicate Registers Specifies the removal of duplicate registers.

When set to True (default), LSE removes a register if it is identical to another register. If two registers generate the same logic, the second one will be deleted and the first one will be made to fan out to the second one's destinations. LSE will not remove duplicate registers if this option is set to False.

This option is equivalent to the “-remove_duplicate_regs” option in the SYNTHESIS command.

Remove LOC Properties (for LSE) Setting this to On removes LOC properties in the synthesized design before building the Native Generic Database (.ngd) file.

Resolved Mixed Drivers (for LSE) If a net is driven by a VCC or GND and active drivers, setting this option to True connects the net to the VCC or GND driver.

Resource Sharing (for LSE) When this is set to True (default), the synthesis tool uses resource sharing techniques to optimize area.

With resource sharing, synthesis uses the same arithmetic operators for mutually exclusive statements; for example, with the branches of a case statement. Conversely, you can improve timing by disabling resource sharing, but at the expense of increased area.

This option is equivalent to the “-resource_sharing” option in the SYNTHESIS command.

ROM Style Allows you to globally implement ROM architectures using dedicated, distributed ROM, or a combination of the two (Auto).

This applies the syn_romstyle attribute globally to the design by adding the attribute to the module or entity. You can also specify this attribute on a single module or ROM instance.

Specifying a syn_romstyle attribute globally or on a module or ROM instance with a value of:

- Auto (default) – Allows the synthesis tool to choose the best implementation to meet the design requirements for speed, size, and so on.
- Logic – Causes the ROM to be implemented using the distributed ROM or PFU resources. Specifically, the logic value will implement ROM to logic (LUT4) or ROM technology primitives (such as ROM16X1, ROM32X1, ROM64X1, and so on).
- EBR – Causes the ROM to be mapped to dedicated EBR block resources. ROM address or data should be registered to map it to an EBR block. If your ROM resources are limited, for whatever reason, you can map

additional ROM to registers instead of the dedicated or distributed RAM resources using this attribute.

Infer ROM architectures using a CASE statement in your code. For the synthesis tool to implement a ROM, at least half of the available addresses in the CASE statement must be assigned a value. For example, consider a ROM with six address bits (64 unique addresses). The CASE statement for this ROM must specify values for at least 32 of the available addresses.

This option is equivalent to the “-romstyle” option in the SYNTHESIS command.

Target Frequency Specifies the target frequency setting. This frequency applies to all the clocks in the design. If there are some clocks defined in an .ldc file, the remaining clocks will get this frequency setting. When a create_clock constraint is available in an .ldc file, LSE ignores the Target Frequency setting for that clock and uses the value from the create_clock constraint instead.

This option is equivalent to the “-frequency” option in the SYNTHESIS command.

Use Carry Chain Turns on (True) or off (False) carry chain implementation for adders. Default is True.

This option is equivalent to the “-use_carry_chain” option in the SYNTHESIS command.

Use IO Insertion When set to True, LSE uses I/O insertion and GSR.

This option is equivalent to the “-use_io_insertion” option in the SYNTHESIS command.

Use IO Registers When True, this option forces the synthesis tool to pack all input and output registers into I/O pad cells based on the timing requirements for the target device family. Auto, the default setting, enables this register packing if Optimization Goal is set to Area. If Optimization Goal is Timing or Balanced, Auto disables register packing.

This option is equivalent to the “-use_io_reg” option in the SYNTHESIS command.

You can also control packing on individual registers and ports. See “syn_useioff” on page 1398.

Use LPF Created from SDC in Project (for LSE) LSE creates a preference (.lpf) file based on the Synopsys Design Constraint (.sdc) file. (When you use LSE, SDC constraints must be in a Lattice Design Constraints (.ldc) file.) When this option is set to True, the synthesis constraints are also applied to the Map Design stage of implementation.

VHDL 2008 (for LSE) When this is set to True, VHDL 2008 is selected as the VHDL standard for the project.

Translate Design Options

This page lists all the strategy options associated with the Translate Design process.

- Command Line Options (for Synplify Pro Translate Design)
- Consistent Bus Name Conversion
- Hardware Evaluation
- Macro Search Path
- Memory Initial Value File Search Path
- “Partition Search Path” on page 1139
- Remove LOC Properties

Command Line Options (for Synplify Pro Translate Design) Enables EDIF2NGD command line options for the Synplify Translate Design process.

To enter a command line option:

1. In the Strategy dialog box, select **Translate Design** in the Process list.
2. Double-click the Value column for the Command Line Options option.
3. Type in the option and its value (if any) in the text box.
4. Click **Apply**.

For detailed descriptions of EDIF2NGD netlist translation command line options, see “Running EDIF2NGD from the Command Line” on page 2487.

Consistent Bus Name Conversion Allows you to set preferences for bus signals in Verilog and VHDL designs that can be honored by LSE, Synopsys Synplify Pro, and Mentor Graphics Precision synthesis tools.

Four options are available.

- Synplify – Converts the bus signal names to match the Synplify Pro format. The Verilog or VHDL bus signal preferences will then be honored by Synplify Pro.
- Precision – Converts the bus signal names to match the Precision format. The Verilog or VHDL bus signal preferences will then be honored by Precision RTL Synthesis.
- Lattice – Converts “( )”, “[ ]”, “< >”, “{ }” in the bus signal names to underscores “_”. For example, the bus name aq[i] is converted to aq_i_, bq(i) is converted to bq_i_, cq<i> is converted to cq_i_, dq{i} is converted to dq_i_, and eq_i to eq_i_.
- None (default) – Does not convert the bus signal names. The Verilog or VHDL bus signal preferences set for Synplify Pro will not be honored by Precision, and vice versa.

Hardware Evaluation (for Translate Design) Enables or disables the ability to temporarily test IP in a device without an IP license. If enabled, a timer is added to the design that allows unlicensed IP to function for about 4 hours in a device. If disabled, you cannot generate a bitstream if there are any unlicensed IP in the design.

You might want to disable this option to refine your design while waiting for the license. You will not be able to generate a bitstream, but you will be able to see how resources are used (without the timer) and close timing. When you get the license, you can then generate the bitstream.

Regardless of how this option is set, if there are any unlicensed IP in the design, some features of Diamond, such as gate level simulation and EPIC, are blocked.

Macro Search Path (for Translate Design) Allows you to specify a path (or paths) to locate physical macro files used in a given design. The software will add the specified paths to the list of directories to search when resolving file references.

You don't need to specify a search path if the necessary .ngo or .nmc file is in the directory containing the top-level .ngo file or if the FILE attribute in the design gives a complete path name for the file (instead of a relative path name).

The software follows the below order to search for .ngo files.

1. Current implementation directory
2. Project directory
3. Directories where the LPC or IPX source files reside
4. User-specified macro search paths

To specify a macro search path, double-click the Value box, and directly enter the path or click the **...** button to browse for one or more paths.

Memory Initial Value File Search Path (for Translate Design) Allows you to specify paths to locate memory initialization files (.mem) used in a given design. Diamond will add the specified paths to the list of directories to search when resolving file references.

To specify a search path, double-click the Value box and directly enter the path or click the **...** button to browse for the path.

Partition Search Path Allows you to specify paths to locate native generic object files (.ngo) that define partitions used in the design. Diamond will add the specified paths to the list of directories to search when resolving file references. The partition .ngo files will be loaded and used for replacing the original partition.

To specify a search path, double-click the Value box and directly enter the path or click the **...** button to browse for the path.

Remove LOC Properties (for Translate Design) Setting this to On removes LOC properties in the synthesized design before building the Native Generic Database (.ngd) file.

Map Design Options

This page lists all strategy options associated with the Map Design process. The options available for user setting are dependent on the target device of your project.

- Command Line Options
- Ignore Preference Errors
- Infer GSR
- IO Registering
- Overmap Device if Design Does Not Fit
- Pack Logic Block Util.
- Register Retiming
- Report Signal Cross Reference
- Report Symbol Cross Reference
- Timing Driven Mapping
- Timing Driven Node Replication
- Timing Driven Packing

Command Line Options (for Map Design) Enables additional command line options for the associated process.

To enter a command line option:

1. In the Strategy dialog box, select the associated process in the Process list.
2. Double-click the Value column for the Command Line Options option.
3. Type in the option and its value (if any) in the text box. For example: -exp parPathBased=ON
4. Click **Apply**.

To reference more information about command line options for the Map Design process, type map -h <architecture> in a command line window. For detailed options description, refer to Running MAP from the Command Line.

Ignore Preference Errors (for Map Design) Enables or disables the Map Design process to ignore errors in the preference file.

With the default setting of True, Map Design ignores errors in the preference file and continues processing. When you set it to False, Map Design terminates processing and issues an error message.

The Map report includes the setting of this option and the preference errors found.

Infer GSR Enables or disables the GSR inferencing.

The default is True. By default, if the input design (NGD) does not include a GSR buffer, MAP will infer one based on the signal that drives the most set/reset loads.

If this is set to False, all the GSR_NET preferences in the logical preference file (LPF) will be ignored.

IO Registering Directs the I/O register packing.

Valid options are:

- Input – Packs the input registers only.
- Output – Packs the output registers only.
- Both – Packs both the input and output registers whenever possible.
- None – Does not pack the IO registers but uses PFF/PFU resources only.
- Auto (default) – Allows the software to decide whether to pack IO registers or not.

This option overrides the register types of the native generic database (NGD). The USE DIN or USE DOUT preference in the logical preference file (LPF) overrides this option.

Overmap Device if Design Does Not Fit When this is set to True, the Map Design process will write a Native Circuit Description (.ncd) file even if the device is too small for the design. This .ncd file cannot be used for Place and Route Design, just for you to observe mapping results.

Pack Logic Block Util. Sets the relative density (of available slices) at which the slices within a device are to be packed, in terms of a percentage of the available slices in the device.

This option has great control of the packing density. The value range is 0-100 percent (100 = minimum packing; 0 = maximum density).

If this option is not specified (blank), it defaults to 97 percent. The result will be a less dense packing, depending on the size of the design relative to the number of available slices in the device. If the design is large compared with the number of available slices in the device, the mapper will make a reasonable effort to pack the design so that it fits in the device.

If you specify a density value for this option, the mapper will attempt to pack the device to that density. The “0” setting results in the densest mapping. If the design is large compared with the target density, the mapper will make an aggressive packing effort to meet your target. However, this may adversely impact the design’s f_{MAX} performance.

Register Retiming Moves registers across combinational logic to balance timing according to the constraints t_{SU} (INPUT_SETUP), t_{CO} (CLOCK_TO_OUT), and f_{MAX} (FREQUENCY).

This is a typical logic optimization technique to balance combinational logic across register-pairs to allow you to maximize clock frequency. However, there is no guarantee that a better f_{MAX} can be achieved. The f_{MAX} constraint activates re-timing around all registers. The t_{SU} and t_{CO} constraints may de-activate re-timing on I/O registers depending on the balancing of t_{SU} versus f_{MAX} and t_{CO} versus f_{MAX}. The re-timing process stops when it estimates that the constraints have been met.

Register Retiming is turned off by default and should not be used on the initial run until you have determined whether or not it could be used to optimize your design to meet timing.

For more information, see Map Register Retiming.

Report Signal Cross Reference When this is set to True, the map report (.mrp) will show where nets in the logical design were mapped in the physical design (NCD).

The default is False.

Report Symbol Cross Reference When this is set to True, the map report (.mrp) will show where symbols in the logical design were mapped in the physical design (NCD).

The default is False.

Timing Driven Mapping Allows you to apply timing-driven logic collapsing and optimization to further optimize the critical paths.

Timing Driven Mapping reads the preference file and calculates the slacks for all constrained paths. The mapping optimizes the critical paths based on the slack distributions. Options are True and False (default).

Timing Driven Node Replication Allows you to activate timing driven logic replication.

When this is set to True, the software replicates a LUT-4 that has multiple fanouts to flip-flops (FFs), adding LUT for each FF if the LUT belongs to the timing path. This allows packing LUT/FF in the same slice for all FFs. Default is False.

Timing Driven Packing Allows you to activate timing driven packing of LUT/FF, FF/LUT, and LUT/LUT in the same slice.

Options are True and False (default).

Map Trace Options

This page lists all strategy options associated with the Map Trace process.

- Analysis Options
- Auto Timing
- Check Unconstrained Connections (for Map Trace)
- Check Unconstrained Paths
- Full Name
- Number of Unconstrained Paths (for Map Trace)
- Report Asynchronous Timing Loops
- Report Style
- Route Estimation Algorithm (%)
- Worst-Case Paths (0-4096)

Analysis Options (for Map Trace) Specifies the analysis type.

- Hold Analysis – Performs hold analysis.
- Standard Setup Analysis (default) – Performs setup time checks on the following preferences: FREQUENCY, CLOCK_TO_OUT, and INPUT_SETUP.
- Standard Setup with Hold Analysis on IOs – Performs standard setup analysis and a hold time analysis on I/Os in the same report file. Applies to INPUT_SETUP and CLOCK_TO_OUT timing preferences.
- Standard Setup and Hold Analysis – Performs both the Standard Setup Analysis and the Hold Analysis.

Auto Timing (for Map Trace) Lists the auto-generated timing preferences in the TRACE report.

If you do not define any timing preference for your FPGA design, Lattice Diamond will automatically generate timing preferences for you. Set this to True (default) if you want your TRACE report to list the auto-generated timing preferences.

Check Unconstrained Connections (for Map Trace) Reports connections not covered by a timing preference.

Check Unconstrained Paths (for Map Trace) Reports paths not covered by a timing preference.

Full Name (for Map Trace) When this is set to True, the TRACE report will show full-length component names instead of the truncated names.

Number of Unconstrained Paths (for Map Trace) Sets the maximum number of unconstrained paths that will be reported. "Unconstrained paths" are those not covered by a timing preference.

Report Asynchronous Timing Loops (for Map Trace) Produces a report of paths that could not be analyzed because they are asynchronous loop.

Report Style (for Map Trace) Specifies the type of the timing report.

- Verbose Timing Report (default) – Lists detailed logic and route delays for all constrained paths and nets in the design.
- Error Timing Report – Lists logic and route delays for each path and net that causes a timing error.

Route Estimation Algorithm (%) Configures the route estimation algorithm for pre-route static timing analysis.

- 0 (default) – Enables route delay estimation based on a suite of Lattice benchmark designs. Note that delay estimates are based on the default performance grade of the target device.
- 1~100 – Specifies logic delay as a percentage of the overall path delay where the total delay is the sum of logic and route delays.

Worst-Case Paths (0-4096) (for Map Trace) Specifies the number of paths to be reported for each timing preference. Enter zero or a grater value.

- When **Verbose Timing Report** is selected, the value you specify limits the number of paths reported for each timing preference (0 = no limit). There is no practical limit, although unlimited reporting may exhaust available memory and disk space.

  Scoring a path refers to the timing extraction performed by the static timing analysis engine. TRACE will always extract the worst case paths, however, you can choose to analyze and score more than the system default.

  Given a large design with many paths to examine, you may wish to limit the report size by using more specific preferences or BLOCK type exceptions temporarily to mask paths that are not of interest.
- When **Error Timing Report** is selected, the specified number of paths with timing error will be printed out. If you enter **0**, no path will be reported.

Place & Route Design Options

This page lists all strategy options associated with the Place & Route Design process. The options available for user setting are dependent on the target device of your project.

- Auto Hold-Time Correction
- Clock Skew Minimization
- Command Line Options
- Congestion-Driven Placement
- Congestion-Driven Routing
- Create Delay Statistic File
- Default Clock Frequency
- Disable Timing Driven
- Generate TRACE Report for Each Iteration

- Ignore Preference Errors
- Multi-Tasking Node List
- Path-based Placement
- Placement Effort Level
- Placement Iteration Start Pt
- Placement Iterations
- Placement Save Best Run
- Placement Sort Best Run
- Remove Previous Design Directory
- Routing Delay Reduction Passes
- Routing Method
- Routing Passes
- Routing Resource Optimization
- Run Placement Only
- Run Time Reduction
- SLVS MIPI LVCMOS Output
- Stop Once Timing is Met

Auto Hold-Time Correction Controls whether the router will automatically insert extra wires to compensate for hold-time violations on affected registers. Auto hold-time correction has a negative impact on your setup time. If the setup time is more design critical, turn off this option. The default is On.

Note

If there is a setup timing error, auto hold-time correction will not be done. If you want the correction to be done anyway, add the following option to “Command Line Options” (see below) in the Place & Route Design strategy:

```
-exp parHold=1
```

See parHold for more information.

There are several `par -exp` command line options that affect auto hold-time correction. All of these options can be added to “Command Line Options” (see below) in the Place & Route Design strategy.

- `parHold=1` forces correction to run.
- `parHoldIter=<value>` sets the maximum number of iterations to run.
- `parHoldLimit=<value>` sets the maximum number of hold time violations to be processed.
- `parHoldOnly=1` causes place and route to be run for auto hold time correction only. Use with parHold.
- `parHoldSpeedGrade=<value>` sets the performance grade to be used.

For details, see Running PAR from the Command Line.

For more about using this option, see:

- "Hold-Time Error Correction" on page 605
- "Changing Temperature and Voltage for Auto Hold-Time Correction" on page 608.

Clock Skew Minimization Balances routing to reduce skews for clock signals that are not assigned to the global clock tree.

Options are Off (default), 1, and 2.

- The 1 setting routes from driver pins to each clock load with pre-computed delay lower bound for balanced routing. This setting may help when the candidate net has a very small span (for example, the bounding box span is within 2 to 3 PLCs) and possesses small skew.
- The 2 setting routes a clock trunk first and then routes from the trunk to each clock load with pre-computed delay lower bound for balanced routing. This setting usually creates better results with less time and memory use.

Command Line Options (for Place & Route Design) Allows you to specify options from the par command without directly using the command line. Type in a string of options without the par command. For example: `-exp parPathBased=ON:parHold=1`

For detailed descriptions of placement, routing, and PAR explorer (-exp) command line options, see Running PAR from the Command Line.

Congestion-Driven Placement Enables or disables the congestion-driven placement algorithm.

The available options are:

- 1 – Enables the congestion-driven placement algorithm.
- 0 – Disables the congestion-driven placement algorithm.
- Auto – Automatically turns on or off the congestion-driven placement based on the device size and design parameters.

The default is Auto for LatticeECP2/M, LatticeECP3, and LatticeXP2 devices, and 0 for other devices.

Congestion-driven placement is compatible with all Lattice FPGA device families. It is most beneficial for designs targeted to LatticeSC/M, LatticeECP2/M, LatticeECP3, and LatticeXP2 device families.

See Routing Method and Congestion-Driven Routing for more congestion-driven options.

Congestion-Driven Routing Enables or disables the congestion-driven routing algorithm.

The available options are:

- 1 – Enables the congestion-driven routing algorithm.
- 0 – Disables the congestion-driven routing algorithm.
- Auto – Automatically turns on or off the congestion-driven routing based on the device size and design parameters.

The default is 1 for LatticeECP2/M, LatticeECP3, and LatticeXP2 devices, and 0 for other devices.

Congestion-driven options like Congestion-Driven Placement and Congestion-Driven Routing can improve performance given a design with multiple congestion “hotspots.” The Layer > Congestion command of the Floorplan View can help visualize routing congestion. Large congested areas may prevent the options from finding a successful solution.

Congestion-driven routing is compatible with all Lattice FPGA device families. It is most beneficial for designs targeted to LatticeECP2/M, LatticeECP3, and LatticeXP2 device families.

See Routing Method and Congestion-Driven Placement for more congestion-driven options.

Create Delay Statistic File When set to True, outputs a delay report file (.dly) on each PAR run.

The delay report file contains delay information for each net in the design for each run. You can generate this file when you use the -y option in the command line.

Default Clock Frequency (MachXO3L/LF 9400C/E devices only.) The proper Default Clock Frequency (MHz) must be set before running the Thermal Analysis process. Default setting of 0 will cause an error.

Disable Timing Driven Enables or disables the timing-driven option for the PAR run.

When this is set to True, the timing-driven option for the PAR run will not be used. If this is set to False, PAR automatically uses the timing-driven option if the Timing Wizard is present and if any timing preferences are found in the preference file. If selected, the timing-driven option is not invoked in any case and cost-based placement and routing are done instead.

Two examples of situations in which you might disable this option are:

- You have timing preferences specified in your preference file, but you want to execute a quick PAR run without using the timing-driven option to give you a rough idea of how difficult the design is to place and route.
- You only have a single license for the timing-driven option but you want to use this license for another application (for example, to perform timing-driven routing within EPIC) that will run at the same time as PAR. This option keeps the license free for the other application.

Generate TRACE Report for Each Iteration Generates a post-PAR TRACE report (.twr) for each intermediate NCD created by multi-PAR.

Ignore Preference Errors (for Place & Route Design) Enables or disables the Place and Route Design process to ignore errors in the preference file.

With the default setting of True, the PAR process ignores errors in the preference file and continues processing. When you set it to False, the PAR process terminates processing and issues an error message.

The Place & Route report includes the setting of this option and the preference errors found.

Multi-Tasking Node List Allows you to specify the node file name for the multi-tasking PAR.

The multi-tasking PAR allows you to use multiple machines (nodes) that are networked together for a multi-run PAR job, significantly reducing the total amount of time for completion.

For more information on multi-tasking PAR, see Running Multiple PAR Jobs in Parallel.

Path-based Placement Allows you to apply path-based placement. Path-based placement gives better performance and more predictable results.

Options are Off (default) and On.

Placement Effort Level Specifies the effort level of the design from 1 (simplest designs) to 5 (most complex designs).

The level is not an absolute; it shows instead relative effort. After you use PLACE & ROUTE for a while, you will be better able to estimate whether a design is simple or complex. If you place and route a simple design at a complex level, the design will be placed and routed properly, but the process will take more time than placing and routing at a simpler level. If you place and route a complex design at a simple level, the design may not route to completion or may route less completely (or with worse delay characteristics) than at a more complex level.

Placement Iteration Start Pt Specifies the cost table to use (from 1-100) to begin the PAR run.

The default is 1. Cost tables are not an ordered set. There is no correlation between a cost table's number and its relative value. If cost table 100 is reached, placement does not begin at 1 again, even if command options specify that more placements should be performed.

Placement Iterations Specifies the maximum number of placement/routing passes (0-100, 0 = run until solved) to be run (regardless of whether they complete) at the Placement Effort Level.

Each iteration uses a different cost table when the design is placed and will produce a different Native Circuit Description (.ncd) file. If you specify a Starting Cost Table, the iterations begin at that table number.

Placement Save Best Run Determines the number (1-100) of best outputs of the Place and Route run to save (defaults to 1).

If no number is specified, all output designs produced by the PLACE & ROUTE run are saved. The best outputs are determined by a scoring system described in the section titled Scoring the Routed Design.

This option does not care how many iterations you performed or how many effort levels were used. It compares every result to every other result and leaves you with the best number of Native Circuit Description (.ncd) files.

Placement Sort Best Run Specifies how to sort the results of the Place and Route run in the PAR report. Results are sorted first by the number of unrouted connections. Then the results are ranked by timing. Choose whether you want to use the worst-slack value or the timing score to rank the results.

The default is Worst Slack.

Remove Previous Design Directory When this is set to True (default), the software removes the contents of the project design directory before Place and Route Design is rerun.

This prevents the accumulation of files from multiple Place & Route Design processes in the same directory.

Routing Delay Reduction Passes Determines the number of passes to be run for Routing Delay Reduction and Routing Resource Optimization properties. The first delay-based cleanup pass produces the greatest improvement.

The value range is 0 to 100.

Routing Method Specifies router algorithm.

Options are Default (rip-up-based routing) and NBR (negotiation-based routing).

NBR is supported by LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP2, and LatticeSC/M device families. The Routing Method option defaults to NBR for LatticeECP2, LatticeECP3, and LatticeXP2 devices, and Default for the other supported devices.

NBR involves an iterative routing algorithm that routes connections to achieve minimum delay cost. It does so by computing the demand on each routing resource and applying cost values per node. It will complete when an optimal solution is arrived at or the number of iterations is reached. Compared with the traditional rip-up-based routing, NBR can provide better results in performance (f_{MAX}), CPU time and completion rate.

A combination of this option and the Congestion-Driven Placement and Congestion-Driven Routing options can yield improved results. The relative runtime and quality by option is:

- Routing Method = Default, Congestion-Driven Placement = 0, Congestion-Driven Routing = 0: low congestion
- Routing Method = Default, Congestion-Driven Placement = 1, Congestion-Driven Routing = 1: moderate congestion
- Routing Method = NBR, Congestion-Driven Placement = 1: very congested

Routing Passes Determines the maximum number of routing passes the place and route tool takes.

One routing pass is an attempt to route each unrouted connection. While most designs route completely after a single routing pass, a design that is tightly packed or that has very tight timing requirements may require multiple routing passes to complete. If Routing Passes is not set, the router will intelligently decide the number of routing passes to make.

Routing Resource Optimization Determines the number of cost-based cleanup passes to run.

If not used, the router runs one cost-based cleanup pass. If you run both cost-based and delay-based cleanup passes, the cost-based passes run before the delay-based passes.

Run Placement Only Setting this to True prevents the design from being routed. PAR will output a placed, but not routed Native Circuit Description (.ncd) file.

This option defaults to False.

Run Time Reduction This is a multi-seed PAR option that uses checkpoints for comparing the result of the current placement seed with the results already obtained. A placement seed that is not producing better results will automatically be terminated.

For more information, refer to “Run Time Reduction” on page 609/

This option defaults to True.

SLVS MIPI LVCMOS Output This option allows the user to select between LVCMO12_18 ONLY (default) or LVCMOS_NOT_PERMITTED.

LVCMO12_18 ONLY - LVCMOS output is limited to LVCMOS12 or LVCMOS18. I/O placement is limited by LCT plus.

- SLVS/MIPI bank: Only allows SLVS input, MIPI input/bidi, LVCMO12 output or LVDS output when bank VCCIO is 2.5 (if only using SLVS in the bank).
- Neighbor bank: Only allows LVCMOS12, LVCMOS18 or LVDS output.

LVCMOS_NOT_PERMITTED - No LVCMOS output is allowed. I/O placement is limited by LCT plus.

- SLVS/MIPI bank: Only supports either SLVS input or MIPI input/bidi; no any other IO_TYPE output is allowed.
- Neighbor bank: Only supports LVDS output.

Stop Once Timing is Met Setting this to True forces the Place and Route Design process to stop as soon as the timing requirement is satisfied. This option has no effect if the "Generate TRACE report for each iteration" option is set to True or if using Run Manager to produce multiple place-and-route runs.

Place & Route Trace Options

This page lists all strategy options associated with the Place & Route Trace process.

- Analysis Options
- Auto Timing
- "Check Unconstrained Connections (for Place & Route Trace)" on page 1152
- Check Unconstrained Paths
- Full Name
- "Number of Unconstrained Paths (for Place & Route Trace)" on page 1152
- Report Asynchronous Timing Loops
- Report Style
- Speed for Hold Analysis
- Speed for Setup Analysis
- Worst-Case Paths (0-4096)

Analysis Options (for Place & Route Trace) Specifies the analysis type.

- Hold Analysis – Performs hold analysis. For post-route analysis, you may override the default performance grade (the performance grade of the target device) using the Speed for Hold Analysis option.
- Standard Setup Analysis (default) – Performs setup time checks on the following preferences: FREQUENCY, CLOCK_TO_OUT, and INPUT_SETUP. For post-route analysis, you may override the default performance grade (the performance grade of the target device) using the Speed for Setup Analysis option.
- Standard Setup with Hold Analysis on IOs – Performs standard setup analysis and a hold time analysis on I/Os in the same report file. Applies to INPUT_SETUP and CLOCK_TO_OUT timing preferences. For post-route analysis, you may override the default performance grade (the performance grade of the target device) using the Speed for Setup Analysis option.

- Standard Setup and Hold Analysis – Performs both the Standard Setup Analysis and the Hold Analysis.

Auto Timing (for Place & Route Trace) Lists the auto-generated timing preferences in the TRACE report.

If you do not define any timing preference for your FPGA design, Lattice Diamond will automatically generate timing preferences for you. Set this to True (default) if you want your TRACE report to list the auto-generated timing preferences.

Check Unconstrained Connections (for Place & Route Trace) Reports connections not covered by a timing preference.

Check Unconstrained Paths (for Place & Route Trace) Reports paths not covered by a timing preference.

Full Name (for Place & Route Trace) When this is set to True, the TRACE report will show full-length component names instead of the truncated names.

Number of Unconstrained Paths (for Place & Route Trace) Sets the maximum number of unconstrained paths that will be reported. "Unconstrained paths" are those not covered by a timing preference.

Report Asynchronous Timing Loops (for Place & Route Trace) Produces a report of paths that could not be analyzed because they are asynchronous loop.

Report Style (for Place & Route Trace) Specifies the type of the timing report.

- Verbose Timing Report (default) – Lists detailed logic and route delays for all constrained paths and nets in the design.
- Error Timing Report – Lists logic and route delays for each path and net that causes a timing error.

Speed for Hold Analysis Specifies performance grade for hold analysis. This option allows you to override the default m (minimum) performance grade for hold time analysis, which represents faster silicon than the fastest performance grade of the device being targeted.

Speed for Setup Analysis Specifies performance grade for setup analysis. This option allows you to override the default performance grade which runs setup analysis against the performance grade of the device currently targeted by the project implementation.

Worst-Case Paths (0-4096) (for Place & Route Trace) Specifies the number of paths to be reported for each timing preference. Enter zero or a grater value.

- When **Verbose Timing Report** is selected, the value you specify limits the number of paths reported for each timing preference (0 = no limit). There is no practical limit, although unlimited reporting may exhaust available memory and disk space.

Scoring a path refers to the timing extraction performed by the static timing analysis engine. TRACE will always extract the worst case paths, however, you can choose to analyze and score more than the system default.

Given a large design with many paths to examine, you may wish to limit the report size by using more specific preferences or BLOCK type exceptions temporarily to mask paths that are not of interest.

- When **Error Timing Report** is selected, the specified number of paths with timing error will be printed out. If you enter 0, no path will be reported.

IO Timing Analysis Options

The following option is associated with the IO Timing Analysis process.

All Performance Grade Controls whether the I/O timing report (.ior) will give an in-depth analysis on all available performance grades or will just produce a summary report on the worst-case performance grade.

- True - The I/O Timing Report will summarize the worst-case scenario of all available performance grades.
- False (default) - The I/O Timing Report will only contain a summary of the worst-case performance grade for the given device.

Timing Simulation Options

This page lists all strategy options associated with the Timing Simulation process. The options available for user setting are dependent on the target device of your project.

- Generate PUR in the Netlist
- Generate X for Setup/Hold Violation
- Hold Check Min Speed Grade
- Multichip Module Prefix
- Negative Setup-Hold Times
- Retarget Speed Grade
- Timing Simulation Max Delay between Buffers (ps)
- Transport Mode of Path Delay in VHDL
- Verilog Hierarchy Separator
- Write Verbose Netlist

Generate PUR in the Netlist When this is set to False, the timing simulation file generation process will not write PUR instance in the Verilog/ VHDL backannotation netlist. Then you have to instantiate PUR in the test bench.

Generate X for Setup/Hold Violation When this is set to True, the Timing Simulation process will place X notifiers in the output file on flip-flops with setup and/or hold time violations.

Hold Check Min Speed Grade Setting this to True replaces all timing information for back annotation with the minimum timing for all paths.

This option is used for simulation of hold time requirements. Separate simulations are required for hold time verification (-min switch) and delay time verification (normal output).

Timing Simulation Max Delay between Buffers (ps) Distributes routing delays by splitting the signal and inserting buffers. The delay value assigned represents the maximum delay number in picoseconds between each buffer (1000 ps by default).

Multichip Module Prefix Adds a prefix to module names to make them unique for multi-chip simulation.

Negative Setup-Hold Times Allows you to select negative setup time and negative hold time for better accuracy.

The default is True. You can set it to False for those simulators that might not be able to handle negative setup- hold times.

Retarget Speed Grade Retargets back annotation to a different performance grade than the one used to create the Native Circuit Description (.ncd) file.

You are limited to those performance grades available for the device used in the NCD file.

Transport Mode of Path Delay in VHDL When this is set to True, the VitalTransport glitch propagation type is used rather than the default OnDetect glitch propagation type. This selection should be used to prevent filtering of a glitch when an input pulse is shorter than the propagation delay..

The default is False. The option is only available for VHDL designs.

Verilog Hierarchy Separator Specifies the hierarchy separator character which will be used in name generation when the design hierarchy is flattened.

The default setting is “/”. You can specify an alpha-numeric character or special character as the hierarchy separator.

For alpha-numeric characters, just enter the character as is in the edit box. For special characters (such as +, -, and so on), encapsulate the character with double quotes, for example, “+”, “-”.

The option is only available for Verilog designs.

Write Verbose Netlist When this is set to False (default), a parameterized netlist will be generated. The parameterized format contains less details, but is more compact in size and memory efficient within the simulator.

Bitstream Options

This page lists all strategy options associated with the bitstream generation process. The options available for user setting are dependent on the target device of your project.

- Address
- Byte Wide Bit Mirror
- Chain Mode
- Create Bit File
- Disable UES
- Enable I/O in Re-configuration
- Enable Timing Check
- External Clock
- Grant Timeout
- LengthBits
- No Header
- Output Format
- Output Zero Frames
- PROM Data Output Format
- ReadBack
- ReadCapture
- Register Configuration
- Reset Config RAM in Re-configuration
- Run DRC
- Search Path
- SPI Start Address
- Startup Clock
- Sysbus Clock Config
- Sysbus Config
- UFM Initval Order
- Wait State Timeout
- XRES Calibration Control

Address Specifies the bitstream addressing mode.

- Increment (default) – For general use. Contains data information but no address information; each frame is loaded from lowest address to highest address.

- Explicit – For testing and debugging. Requires address information followed by data information, one address frame or more (depending on how many locations must be written).

Byte Wide Bit Mirror Reverses the order of the bits on a byte-by-byte basis.

Chain Mode When enabled, allows daisy chaining with other devices.

- Disabled – The device is not daisy chained with other devices.
- Flowthrough – This option, which can be implemented with either master or slave parallel configuration, pulls the CSON pin low when the device has completed its configuration. The Flowthrough option drives out a static low signal on the CSON pin. It then tristates the device D[0:7] and BUSY pins when configuration is completed so they will not interfere with the next daisy-chained device to be configured. Once Flowthrough mode starts, the device remains in Flowthrough mode until the wake-up sequence is completed. One option for getting out of Flowthrough mode is to toggle CSN and CS1N, which act as reset signals.
- Bypass – This option is used in parallel and serial device daisy chains. When configuration of the device is completed, data coming into the device configuration port overflows serially out of DOUT to the DI of the next slave serial device. In serial configuration mode, the Bypass option connects DI to DOUT by means of a bypass register after configuration is completed. The bypass register is initialized with a 1 at the beginning of configuration. In parallel configuration mode, the Bypass option causes the data incoming from D[0:7] to be serially shifted to DOUT after completion of configuration. The serialized byte-wide register is shifted to DOUT through the bypass register. D0 of the byte wide data is shifted out first, followed by D1, D2, and so on. Once Bypass mode starts, the device remains in Bypass mode until the wake-up sequence finishes. One option for getting out of Bypass mode is to toggle CSN and CS1N, which act as reset signals.

Create Bit File When set to True, generates a bitstream file.

Disable UES Excludes or includes the "User Electronic Signature Data" field in the JEDEC file.

When this is set to True, the generated JEDEC will not contain the “User Electronic Signature Data” field, regardless of what the USERCODE preference is specified in the project preference (.lpf) file. When this is set to False, the “User Electronic Signature Data” field will be written into the JEDEC file.

Enable I/O in Re-configuration When enabled, programs the FPGA to drive outputs (not be tristated) during configuration/reconfiguration.

This option is disabled by default. The only time the “Enable IOs during reconfiguration” option should be used is for partial reconfiguration when it is desired to keep the part operational during reconfiguration, and then extreme care must be taken to ensure that no user outputs are placed at pins required for configuration.

Enable Timing Check When this is set to true and there is timing error in the Place and Route report file, a dialogue box will display before executing bitstream generation.

External Clock Used for testing Master configuration modes.

By default (option No), in Master configuration modes the configuration clock CCLK is an output driven by the internal oscillator (refer to the configuration section of the device data sheet for details). Setting this option to Yes makes CCLK an input.

Grant Timeout Controls an internal system bus grant counter that counts the number of HCLK (sysbus clk) cycles that go by before the grant signal is taken away from the master.

This prevents the system bus from locking up if anything goes wrong. Possible values are 0, 1, 2...15. See Wait State Timeout HCLK cycle values.

LengthBits Specifies bitstream length.

The default bitstream length field is 24 bits. For larger bitstreams, a 32-bit length field may be specified. Daisy chained bitstreams must all use the same size length field.

No Header When this is set to True, the header, or comment string, is not included in the bitstream.

By default, the header is included.

Output Format Specifies the type of bitstream to create for an FPGA device.

The following options are available:

- Bit File (Binary) – Generates a binary configuration file (.bit) that contains the default outputs of the Bit Generation process.
- Raw Bit File (ASCII) – Generates an ASCII raw bit text file (.rbt) of ASCII ones and zeros that represent the bits in the bitstream file. If you are using a microprocessor to configure a single FPGA, you can include the Raw Bit file in the source code as a text file to represent the configuration data. The sequence of characters in the Raw Bit file is the same as the bit sequence that will be written into the FPGA.
- Mask and Readback File (ASCII) – Generates an ASCII bitstream file that is used to compare bit locations and execute a readback of configuration data within an operating FPGA. When you select this option, BITGEN generates a an ASCII mask file (.mska) for the input design and an ASCII readback file (.rbka) for verifying the configuration data.
- Mask and Readback File (Binary) – The mask file is used to compare bit locations and execute a readback of configuration data within an operating FPGA. When you select this option, BITGEN generates a binary mask file (.msk) for the input design and a binary readback file (.rbk) for verifying the configuration data.

Output Zero Frames Controls whether zero frames are written to the output bitstream.

- If you set this option to False, the software skips zero frames and only writes non zero frames to the bitstream. This minimizes the size of the bitstream.

  For example,

  ```
  frame a 10011110011...
  frame c 01011101011...
  ```

  Note

  You don't need to save the zero frames in the bitstream because RAM bits are zeros by default before configuration.

- If this option is set to True (default), the software writes all frames, including zero frames, to the bitstream.

  For example,

  ```
  frame a 10011110011...
  frame b 00000000000...
  frame c 01011101011...
  ```

PROM Data Output Format Specifies the type of PROM data to create for an FPGA device.

You can select Intel Hex 32-bit or Motorola Hex 32-bit as the PROM data format.

ReadBack Controls whether the sysCONFIG readback operation will read from FLASH or SRAM.

Options are FLASH or SRAM. The default is SRAM. If FLASH is selected, then control register 0 bit 31 is set to 1.

ReadCapture When this is enabled, the GLB is targeted during readback.

Register Configuration Controls whether the software will reset all flip-flops before reconfiguring the target FPGA.

If you are performing a partial reconfiguration and don't want to reset all flip-flops, set this option to NoResetDuringReConfig.

- ResetDuringReConfig – Resets all flip-flops before reconfiguration.
- NoResetDuringReConfig – Does not reset flip-flops before reconfiguration.

Reset Config RAM in Re-configuration The Reset option reinitializes the device when you download a bitstream or JEDEC file. The No Reset option retains the current configuration and allows additional bitstream or JEDEC configuration.

Run DRC When this is set to True, the software runs a physical design rule check and saves the output to the Bit Generation report file (.bgn).

Running DRC before a bitstream or JEDEC file is produced will detect any errors that could cause the FPGA to function improperly. If no fatal errors are detected, it will produce a bitstream or JEDEC file. Run DRC is the default.

Search Path Allows you to specify the search path of the auto-configuration file that is located in a directory other than the project directory.

For information on how to use the auto-configuration file, see the LatticeECP2M, LatticeECP3, and LatticeSC/M Data Sheets on the Lattice Web site.

To specify the search path, simply enter the path in the text box. A path that contains spaces should be enclosed in double quotes. If the required auto-configuration files are stored in multiple directories, use the following format.

- In Windows:

  path1;*path2*;*path3*

- In Linux:

  path1:*path2*:*path3*

This option is equivalent to the command line: bitgen -path *path1* -path *path2* ...

SPI Start Address Specifies SPI start address.

The SPI start address is a 6 or 8 digit hex string prefixed with 0x. The default is 0x00000000.

When the device is initially configured using an external SPI Flash memory device, the start address is assumed to be 0x00000000. You can use this option to define an alternate start address that can be stored in a 32-bit register. If the device is re-configured without initialization, the start address specified by the prior bit stream will be used. This allows for multiple configuration images to be stored in the same SPI Flash memory or memories if multiple devices are used.

The user-defined address is latched upon the DONE signal going high so that it does not change during the re-configuration of the device. The memory cells storing the data can be re-written, providing a new start address, but the address used for the current configuration sequence would not change. This eliminates the need to preserve the memory contents by using the RAMCFG option.

Some Flash devices use 24-bit addressing. The software can interpret the 24-bit address and base address. For example, if you specify 24-bit address and base address 1, you should set this option to 0x000001.

Startup Clock Controls whether to use CCLK or a User clock during startup.

The default is CCLK. If a User clock is needed to synchronize startup to a system clock, it must be routed to the FPGA Macro Library element STRTUP.

Sysbus Clock Config Specifies either Reset or NoReset as the system bus clock at reconfiguration.

Sysbus Config Specifies either Reset or NoReset as the system bus at reconfiguration.

UFM Initval Order Sets the order of the TAG memory content of LatticeXP2 devices. Normal is for TAG memories created with Diamond 3.0 or later. Legacy is for TAG memories created with Diamond 2.2 or earlier.

Wait State Timeout Controls an internal system bus timeout counter. The counter counts the number of HCLK (sysbus clk) cycles that go by while the system bus is in wait states.

Valid values are integers from 0 to 15. The default is 5.

- 0 – Forever (never time out)
- 1 – 2^2 HCLK cycles
- 2 – 2^4 HCLK cycles
- 3 – 2^6 HCLK cycles
- 4 – 2^8 HCLK cycles
- 5 – 2^10 HCLK cycles
- 6 – 2^12 HCLK cycles
- 7 – 2^14 HCLK cycles
- 8 – 2^16 HCLK cycles
- 9 – 2^18 HCLK cycles
- 10 – 2^20 HCLK cycles
- 11 – 2^22 HCLK cycles
- 12 – 2^24 HCLK cycles
- 13 – 2^26 HCLK cycles
- 14 – 2^28 HCLK cycles
- 15 – 2^31 HCLK cycles

XRES Calibration Control Controls the calibration circuit connected to XRES pin.

The external resistor combined with an internal calibration control circuit is used to calibrate output drive strength and impedance. The calibration compensates for the effects of variations in process, voltage, and temperature (PVT). It is connected through a 1K+/-1% resistor to ground.

This option is available only for LatticeSC/M designs. Available values are:

- INIT (default) – Allows the calibration circuit to monitor PVT on power-up and during configuration. The circuit latches in the control value and does not allow changes after that. If voltage and temperature does not vary much during operation versus configuration time, INIT is the best choice.
- ON – Allows the calibration circuit to run all the time. If IOPVTCTRL is instantiated in the design, the XRES is controlled by the UPDATE signal. INIT and OFF options are disabled.
- OFF – Turns off the calibration circuit. The result is minimum drive on output buffer, maximum impedance, and maximum termination resistor. OFF is not recommended in most FPGA implementations.

Chapter 16

Hardware How-To

Each of the “How-To” topics that follow describe the high-level process you should follow to accomplish some aspect of creating your FPGA designs. These topics also provide a starting point for you to access all of the available material in the Help system and on the web for the subject matter.

Topics include:

- “How to Design with FPGA Memories” on page 1162
- “How to Use Dedicated DDR Memory Support” on page 1165
- “How to Design with PCS/SERDES” on page 1166
- “How to Design with sysCLOCK PLLs and DLLs” on page 1168
- “How To Design with sysIO Buffers” on page 1171
- “How to Design with sysDSP” on page 1172
- “How to Use the Global Set/Reset (GSR) Signal” on page 1174
- “How to Use the Power Up Set/Reset (PUR) Global Signal” on page 1189
- “How to Use the Tristate Interface (TSALL) Global Signal” on page 1190
- “How to Use the Internal Oscillator” on page 1192

How to Design with FPGA Memories

The Lattice FPGA memory technology helps you easily create and implement a variety of memory organizations including single-port, dual-port, pseudo dual-port RAMs, ROMs, or FIFOs. You can control what FPGA resources will be employed to implement the memory as either sysMEM embedded block RAM (EBR) or distributed RAM across programmable function units (PFUs). The Diamond software helps automate the implementation of memories from design entry through to creation of a programming file.

This How-to topic applies to all Lattice FPGA device families. For FPGA memory feature support by device family, see the references listed at the end of this topic.

Design Flow The following steps describe the typical flow for performing design entry, simulation, synthesis, and place and route of designs that incorporate embedded sysMEM or distributed RAM memories:

1. Open Diamond and choose **File > New > Project** to create a project that targets a supported device family.
2. Click **Tools > IPexpress** to generate a specific configuration of a memory module using the IPexpress module generator. Distributed RAM modules are located in the **Memory_Modules > Distributed_RAM** module folder and embedded sysMEM modules are located in the **Memory_Modules > EBR_Components** folder.
3. In the File List view, double click on your input HDL source file and instantiate one or more memory modules into your HDL design using Diamond's internal Source Editor tool.
4. In the File List view, right click on the Input Files folder. In the popup menu, choose **Add > Existing Source** and in the Add Design File dialog navigate to your <*module_name*>.ipx file to import it into your Diamond project. Alternately, you may import the defining module HDL file as your project source; however, future modifications to the module requires additional steps.

You may bypass IPexpress and directly instantiate memory library primitives into your HDL source as described below.

To instantiate memory library primitives into your HDL source:

1. In the Process view, double click the **Translate Design** process to create a Pre-Map version of the design. A green check mark will indicate if your NGD file was created successfully and the Output view will report results of the run.

   Note

   If you see a yellow triangular icon with an exclamation point, double click on the adjacent warning symbol to view the message in the Output view. Check the Error and Warning tabs in the Output view for potential issues. Generally, use this approach to investigate all warning and error messages in your design flow.

2. In the Process view, double click the **Map Design** process to create a post-map version of the design. Click on the Reports tab in the main view and in the Process Reports folder in the Design Summary list double click Map to open the MRP file. Use the MRP report file to confirm correct parameter usage and the specific device resources utilized.
3. In the Process view, double click the **Place & Route Design** process to create a post-PAR version of the design. Again, in the Process Reports folder in the Design Summary list, double click on Place & Route to open the PAR report file and review the final implementation.
4. You may simulate a project containing a memory module at all phases of the design flow, Functional, Post-Route Functional, or Post-Route Timing Simulation, using your third-party simulator of choice. Diamond allows you to export files for simulation. See applicable documentation in Diamond help for more information.
5. You may bypass IPexpress and directly instantiate memory macros into your HDL source. For more information see the "FPGA Libraries Reference Guide" on page 1665.

Modifying an Existing Memory Block If a configuration or port interface needs to be modified on an existing memory module of your project, use the following guidelines to make changes:

To modify an existing module:

If the <*module_name*>.ipx file is used as the project source in Diamond, double-click the file to open it in IPexpress.

If the HDL file of the module is used as the project source, use the following procedure:

1. In Diamond, choose **Tools > IPexpress** and select the module in the Module list on the left to populate the the macro type, module name, and other relevant text boxes in the module's Configuration tab view in the main area of the view.
2. After you type in the File name and select Module Output click the **Customize** button. A module dialog appears with a schematic block diagram of the module at left and various settings.
3. In the module's Customize dialog, check **Import IPX to Diamond Project** at the bottom left to include the <*module_name*>.ipx file. IPexpress loads the previous configuration.
4. Adjust the module configuration, then click **Generate**. IPexpress overwrites the prior version.

Optimizing Designs that Utilize FPGA Memory Placement of memories to specific physical locations may contribute to improved design performance and density. Design floorplanning of PFU-based memories is typically accomplished using region, bounding box, and HGROUP/UGROUP constraints whereas EBR-based memories are typically placed using simple LOCATE preferences. Using a combination of the Floorplanner and Logical and Physical Preferences with a design iteration strategy can produce a superior implementation.

For more information on design guidelines for optimal results, see the FPGA Design Guide.

References For more information on LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, MachXO, MachXO2, LatticeXP, and LatticeXP2 device memory blocks, refer to the following examples and technical notes from the Lattice Semiconductor Web site:

- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO Devices*
- TN1201, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*

- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1306, *CrossLink Memory Usage Guide*
- FPGA-TN-02066, *Memory Usage Guide for MachXO3D Devices*

See Also ▶"Hardware How-To" on page 1162

How to Use Dedicated DDR Memory Support

This How-to topic applies to all Lattice FPGA device families. More detailed information is available in technical notes on the Lattice web site and in the online Help. These references are listed at the end of this topic.

LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, and LatticeXP/2 devices all support various Double Data Rate (DDR) interfaces using the logic built into the Programmable I/O (PIO). The DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling the performance.

The following steps briefly describe the typical flow for using the DDR I/O to interface to a DDR memory device.

1. Open Diamond and choose **File > New > Project** to create a project that targets a supported device family.
2. In the File List view, double click on your input HDL source file and instantiate the input and output DDR software primitives into your HDL design using Diamond's internal Source Editor tool.

   These primitives provide the 90-degree phase delay of the data strobe (DQS) with respect to the DDR data across process voltage temperature (PVT). They also implement the DDR I/O registers and adjust the system clock polarity to assure correct clock domain transfer in the I/O registers.
3. If you make pin assignments, make sure that the DQ-DQS group assignments are correct. You can use the Spreadsheet view or the HDL file to assign pins.

   Note

   When implementing an input DDR in the ECP/EC20 device family, a potential data corruption might occur in the last read cycle. Please refer to the DQS Postamble section of TN1050, LatticeECP/EC DDR Usage Guide, for a solution to this problem.
4. In the Process view, run the **Map Design** process. This process runs DRC checks to make sure that all the primitives are correctly instantiated.
5. In the Process view, run the **Place & Route Design** process. This process places all the DDR components and checks to see that all the connections made in the HDL are correct.

At any time during the flow you can run pre-route functions or post-route functions to verify the functionality of the DDR interface. You can also run

Timing Simulation after the Place and Route process to verify the interface timing and functionality.

References For more information, refer to the following topics, examples and applications notes from the Lattice Semiconductor web site.

- TN1265, *ECP5 High-Speed I/O Interface*
- TN1050, *LatticeECP/EC and LatticeXP DDR Usage Guide*
- TN1056, *LatticeECP/EC and LatticeXP sysIO Usage Guide*
- TN1099, *LatticeSC DDR/DDR2 SDRAM Memory Interface User's Guide*
- TN1105, *LatticeECP2/M High-Speed I/O Interface*
- TN1180, *LatticeECP3 High-Speed I/O Interface*
- TN1138, *LatticeXP2 High-Speed I/O Interface*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1301, *CrossLink High-Speed I/O Interface*
- FPGA TN-02065, *Implementing High-Speed Interfaces with MachXO3D Device*

See Also ▶"Hardware How-To" on page 1162

How to Design with PCS/SERDES

The Diamond software allows you to design PCS/SERDES solutions for either LatticeSC, LatticeECP2M, and LatticeECP3 device families.

LatticeSC PCS/SERDES The LatticeSC family of FPGA devices combines a high-performance FPGA fabric, up to 32 channels of high-speed SERDES, high performance I/Os and large embedded RAM in a single industry leading architecture. The LatticeSC family of FPGA devices offers features to meet the needs of today's communication network systems. These features include an embedded PCS (Physical Coding Sublayer) supporting advanced system level standards including RapidIO, Sonet, Gigabit Ethernet, 10 Gb Ethernet, Fibre Channel, and PCI Express.

LatticeECP2M PCS/SERDES The LatticeECP2M family of FPGAs combines a high-performance FPGA fabric, high-performance I/Os and large embedded RAM in a single industry leading architecture. All LatticeECP2M devices also feature up to 16 channels of embedded SERDES with associated PCS logic. The PCS logic can be configured to support numerous industry standard high-speed data transfer protocols. Each channel of PCS logic contains dedicated transmit and receive SERDES for high-speed transfers at data rates up to 3.2 Gbps. The PCS logic in each channel can be configured data protocols including Ethernet (1GbE and SGMII), PCI Express, CPRI, and OBSAI.

LatticeECP3 PCS/SERDES The LatticeECP3 family of FPGA devices is optimized to deliver high performance features such as an enhanced DSP architecture, high speed SERDES and high speed source synchronous

interfaces in an economical FPGA fabric. This combination is achieved through advances in device architecture and the use of 65nm technology making the devices suitable for high-volume, high-speed, low-cost applications.

The architecture features high speed SERDES with dedicated PCS functions. High jitter tolerance and low transmit jitter allow the SERDES plus PCS blocks to be configured to support an array of popular data protocols including PCI Express, SMPTE, Ethernet (XAUI, GbE, and SGMII) and CPRI. Transmit Pre-emphasis and Receive Equalization settings make the SERDES suitable for transmission and reception over various forms of media.

Design Flow The Diamond design flow was designed to fuse the embedded portion of the device with the FPGA fabric in a seamless fashion.

The following lists the steps for designing with PCS/SERDES:

- Generate a netlist of the PCS with IPexpress.
- Merge the generated netlist with a supplied example reference design.
- Simulate the merged netlist.
- Synthesize the merged netlist.
- Map, place, and route the merged netlist.
- Perform full timing analysis including back annotated simulation.

Note

The basic design flow steps above will be covered in a tutorial being developed for this feature.

References For more detailed information refer to the following sources available in our help system and from the Lattice Semiconductor web site.

- TN1261 - *ECP5 SERDES/PCS Usage Guide*
- TN1145, *LatticeSC flexiPCS/SERDES Design Guide*
- DS1005, *LatticeSC flexiPCS Data Sheet*
- DS1004, *LatticeSC Family Data Sheet*
- TN1085, *LatticeSC MPI/System Bus*
- TN1124 - *LatticeECP2M SERDES/PCS Usage Guide*
- TN1176 - *LatticeECP3 SERDES/PCS Usage Guide*
- DS1006, *LatticeECP2/M Family Data Sheet*
- DS1021, *LatticeECP3 Data Sheet & Errata*
- TN1114 - *Electrical Recommendations for Lattice SERDES*

See Also ▶"Hardware How-To" on page 1162

How to Design with sysCLOCK PLLs and DLLs

The Lattice sysCLOCK PLL (phase-locked loop) and DLL (delay-locked loop) technology helps you manage clock distribution and skew to improve overall system performance. PLLs are good for applications requiring the lowest output jitter or jitter filtering. DLLs are well suited to applications where the clock may be stopped or transferring jitter from input to output is important. The Diamond software helps automate the implementation of sysCLOCK PLL and DLL modules from design entry through creation of a programming file.

This How-to topic applies to all Lattice FPGA device families. For more sysCLOCK PLL information for these devices, see the references listed at the end of this topic.

Design Flow The following steps describe the typical flow for performing design entry, simulation, synthesis, and place and route of designs that incorporate sysCLOCK PLLs:

1. Open Diamond and choose **File > New > Project** to create a project that targets a supported device family.
2. Click **Tools > IPexpress** to generate a specific configuration of a sysCLOCK PLL or DLL module using the IPexpress module generator.
3. In the File List view, double click on your input HDL source file and instantiate one or more sysCLOCK PLL or DLL modules into your HDL design using Diamond's internal Source Editor tool.
4. Import the <*module_name*>.ipx file to your project's source list using Diamond. Alternatively, you may import the defining module HDL file as your project source; however, future modifications to the module require additional steps.
5. In the Process view, double click the **Translate Design** process to create a pre-map version of the design. A green check mark will indicate if your NGD file was created successfully and the Output view will report results of the run.

   Note

   If you see a yellow triangular icon with an exclamation point, double click on the adjacent warning symbol to view the message in the Output view. Check the Error and Warning tabs in the Output view for potential issues. Generally, use this approach to investigate all warning and error messages in your design flow.

6. In the Process view, double click the **Map Design** process to create a post-map version of the design. Click on the Reports tab in the main view and in the Process Reports folder in the Design Summary list double click Map to open the MRP file. Use the Map Report to confirm correct parameter usage and the specific device resources utilized.
7. In the Process view, double click the **Place & Route Design** process to create a Post-PAR version of the design. Double click Place & Route Report in the Design Summary view to open the PAR report file and review the final implementation.
8. Both phase-locked and delay-locked loops have analog behavior that can not be accurately modeled with most digital simulators, not all functionality

of sysCLOCK PLL hardware is supported in Functional, Post-Route Functional, or Post-Route Timing Simulation phases.

9. You may bypass IPexpress and directly instantiate PLL library primitives into your HDL source.

Important

Note that it may become necessary to regenerate a given PLL or DLL module with updated options because you edited a preference later in the flow (pre- or post-Map) that was a result of an attribute spawned from module generation and was present in the module output file. This will cause the design to become out of synch with the your project's source input module.

For example, it is possible to enhance a FREQUENCY preference in Spreadsheet view or manually in the LPF file in the Diamond flow that was initially auto-generated in the LPF because of clock divider options you selected for module generation in IPexpress. If you updated the FREQUENCY to a higher clock speed that is not in an allowable range for the clock divider options, you will encounter error messages during TRACE timing analysis.

Modifying an Existing sysCLOCK PLL or DLL If a configuration or port interface needs to be modified on an existing sysCLOCK PLL or DLL module of your project, use the following guidelines to make changes using IPexpress:

To modify an existing module:

- If the <*module_name*>.ipx file is used as the project source in Diamond, double-click the file to open it in IPexpress.

If the HDL file of the module is used as the project source, use the following procedure:

1. In Diamond, choose **Tools > IPexpress** and select the module in the Module list on the left to populate the the macro type, module name, and other relevant text boxes in the module's Configuration tab view in the main area of the view.
2. After you type in the File name and select Module Output click the **Customize** button. A module dialog appears with a schematic block diagram of the module at left and various settings.
3. In the module's Customize dialog, check **Import IPX to Diamond Project** at the bottom left to include the <*module_name*>.ipx file. IPexpress loads the previous configuration.
4. Adjust the module configuration, then click **Generate**. IPexpress overwrites the prior version.

Note on Using Tristate Internal Feedback Loops (LatticeECP/EC, LatticeECP2/M, LatticeXP, LatticeSC/M) In cases where you are using a tristate I/O buffer as an internal feedback loop, then the default capacitance

on that buffer should be set to zero. Use the OUTPUT LOAD preference to set this value. See the “Using Tristates as Internal Feedback Loops” section the OUTPUT LOAD preference topic for details.

Note on PLL Output Lock in LatticeSC/M The PLL Output Lock in LatticeSC/M designs requires added logic to prevent false out-of-lock signal.

Depending on certain input conditions of the clock frequency, the relative input clock edge and PLL update edge relationships may cause the digital detect logic of the PLL LOCK signal to transition low for several clock cycles. While the digitally produced lock signal may transition low for several clock cycles, the analog core of the PLL maintains operation within all of its specifications.

This LOCK signal transition may indicate a false out-of-lock occurrence if not properly filtered by the user design. The false out-of-lock signal is due to a synchronization state within the digital lock detect circuit itself and is not indicative of a true, analog out-of-lock condition of the PLL.

To compensate for this potential false indication, it is recommended that you implement a PLL LOCK mask circuit to filter out false PLL out-of-lock (low pulses). Follow the instructions and recommendations provided in the section on Lock Output in technical note, TN1098, “LatticeSC sysCLOCK PLL/DLL User’s Guide” to ensure that you can work around this issue.

References For more information refer to the following examples and application notes from the Lattice Semiconductor web site.

- TN1261 - *ECP5 SERDES/PCS Usage Guide*
- TN1049, *LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide*
- TN1089, *MachXO sysCLOCK PLL Design and Usage Guide*
- TN1098, *LatticeSC sysCLOCK and PLL/DLL User’s Guide*
- TN1103, *LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide*
- TN1178, *LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide*
- TN1261 - *ECP5 SERDES/PCS Usage Guide*
- TN1199, *Lattice MachXO2 sysCLOCK PLL Design and Usage Guide*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1126, *LatticeXP2 sysCLOCK PLL Design and Usage Guide Technical Note*
- TN1304, *CrossLink sysCLOCK PLL/DLL Design and Usage Guide*
- FPGA-TN-02070, *sysClock PLL Usage Guide for MachXO3D*

See Also “Hardware How-To” on page 1162

How To Design with sysIO Buffers

The Lattice sysIO buffer technology helps you easily interface to a variety of advanced system I/O standards. The Diamond software helps automate the implementation of sysIO buffering from design entry through to creation of a programming file.

This How-to topic applies to all Lattice FPGA device families. For specific sysIO information on LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, and LatticeXP/2 device families, see the references listed at the end of this topic.

Design Flow The following steps describe the typical flow for performing design entry, synthesis, and place and route of designs that incorporate sysIO buffers:

1. Open Diamond and create a project that targets a supported device family.
2. Specify sysIO attributes using HDL, the Spreadsheet view, or an ASCII preference file (.prf).
3. If you wish to use HDL-based sysIO attributes, add them to your source code. Then use the Translate Database process to create a Pre-Map version of the design; review the Output view and the Warning and Error tabs that are adjacent to it to confirm that syntax and usage are correct. HDL-based attributes have the advantage of being persistent from revision to revision of your source code; however, Lattice recommends that you avoid adding or modifying preferences with the Spreadsheet view process to avoid contention between preferences specified in both places.
4. If you wish to use the Spreadsheet view to specify sysIO attributes, click **Tools > Spreadsheet** view and use the Pin Attributes tab of the interface to specify each. You can confirm legal combinations sysIO configurations within device PIO by running a PIO DRC design rule check.
5. If you wish, edit the preference file (.prf) using a text editor or Diamond's internal Source Editor to directly to specify sysIO attributes. Double click on the file in the File List view.
6. In the Process view, double click the Map Design process to create a Post-Map version of the design. Double click Map Report in the Design Summary view to open the MRP file to confirm correct sysIO attribute usage and the specific device resources utilized.
7. In the Process view, double click the Place & Route Design process to create a Post-PAR version of the design. Double click Place & Route Report in the Design Summary view to open the PAR report file and review the final implementation.

Modifying a sysIO Buffer on an Existing PIO If a sysIO buffer needs to be modified on an existing PIO of your project, use the following guidelines to make changes:

To modify a sysIO buffer:

- If you defined sysIO preferences within your HDL source, use the internal Source Editor to modify the code and restart the Translate Database process.
- If you defined sysIO preferences using the Spreadsheet view, use it again to modify existing preferences. This bypasses the need to rerun the Translate Database phase of the design flow. Save the new preference file and rerun the Map Design process only.
- If you defined sysIO preferences in an LPF file directly using some text editor, double click the file in the File List view to further modify it using Diamond's internal Source Editor. Save the new preference file and rerun the Map Design process only.

References For more information refer to the following examples and Technical Notes from the Lattice Semiconductor web site.

- TN1262, *ECP5 sysIO Usage Guide*
- TN1056, *LatticeECP/EC sysIO Usage Guide*
- TN1088, *LatticeSC PURESPEED I/O Usage Guide*
- TN1102, *LatticeECP2/M sysIO Usage Guide*
- TN1177, *LatticeECP3 sysIO Usage Guide*
- TN1091, *MachXO sysIO Usage Guide*
- TN1202, *MachXO2 sysIO Usage Guide*
- TN1280, *MachXO3L sysIO Usage Guide*
- TN1305, *CrossLink sysI/O Usage Guide*
- FPGA TN-02065, *Implementing High-Speed Interfaces with MachXO3D Device*

See Also "Hardware How-To" on page 1162

How to Design with sysDSP

The Lattice sysDSP (digital signal processing) technology provides dedicated DSP hardware blocks that perform a variety of high-speed, fixed-point, multiply, sum, and accumulate functions. This technology is supported by the the LatticeECP, LatticeECP2/M, and LatticeXP2 Lattice FPGA device families.

The Diamond software helps automate the implementation of sysDSP block modules from design entry through creation of a programming file. Diamond software also includes Lattice-specific blockset functions for use with the MathWorks Simulink® products for algorithm development.

For more information on sysDSP and the Simulink design flow, see the references at the end of this topic.

Design Flow The following steps describe the typical flow for performing design entry, simulation, synthesis, and place and route of designs that incorporate sysDSP blocks:

1. (Optional) Capture system requirements, physical constraints, and model your sysDSP design using the Simulink design flow. RTL Verilog HDL or VHDL that is compatible with Diamond logic synthesis (Precision or Synplify) is produced by this tool set.
2. Open Diamond and create a project that targets a supported device family.
3. Enter the DSP functionality of your design with any of the following methods:
   - Import Verilog HDL or VHDL source files produced by the MATLAB/ Simulink design flow into your File List using Diamond. Right click on the Input Files folder and choose **Add > Existing Source** and navigate to the file using the Add Design File dialog.
   - Click **Tools > IPexpress** to generate a specific configuration of a sysDSP module using the IPexpress module generator. In the File List view, double click on your input HDL source file and instantiate one or more sysDSP modules into your HDL design using Diamond's internal Source Editor tool. Import the <*module_name*>.ipx file into your project's source list using Diamond. Alternately, you may import the defining module HDL file as your project source; however, modifications to the module requires additional steps.
   - Write HDL that will infer a sysDSP module.
4. In the Process view, double click the Translate Design process to create a Pre-Map version of the design. The Output view will show you if your NGD file was created successfully. Click the adjacent Error and Warning tabs to check for potential issues.
5. In the Process view, double click the Map Design process to create a Post-Map version of the design. Double click Map Report in the Design Summary view to review the DSP block summary section of the MRP file to confirm specific device resources utilized. The Map Design process will check for parameter usage and range violations and report errors in the Error tab adjacent to the Output view.
6. In the Process view, double click the Place & Route Design process to create a Post-PAR version of the design. Double click Place & Route Report in the Design Summary view the DSP block summary section of the PAR file to review the final implementation.
7. You may simulate a project that contains a sysDSP module at all phases of the design flow: Functional, Post-Route Functional, or Post-Route Timing Simulation.
8. You may bypass IPexpress and directly instantiate a sysDSP block library primitives into your HDL source.

See applicable technical notes listed in the References section of this topic for more information about connecting sysDSP software primitives, synthesis style guidelines, module types, modes, and configuration options.

Modifying an Existing ispDSP Module To modify an existing sysDSP module in your project (e.g. change module configuration or port interface), use the following guidelines to make changes:

To modify an existing module:

If the <*module_name*>.ipx file is used as the project source of Diamond, then double-click the file to open it into IPexpress.

If the HDL file of the module is used as the project source, use the following procedure:

1. In Diamond, choose **Tools > IPexpress** and select the module in the Module list on the left to populate the the macro type, module name, and other relevant text boxes in the module's Configuration tab view in the main area of the view.
2. After you type in the File name and select Module Output click the **Customize** button. A module dialog appears with a schematic block diagram of the module at left and various settings.
3. In the module's Customize dialog, check **Import IPX to Diamond Project** at the bottom left to include the <*module_name*>.ipx file. IPexpress loads the previous configuration.
4. Adjust the module configuration, then click **Generate**. IPexpress overwrites the prior version.

Note: If the port interface changes, the code using the module must be modified. If the port interface does not change, no additional modifications must be made, or steps taken once the module has been regenerated.

References For more information refer to the following topics, examples, and application notes from the Lattice Semiconductor web site.

- TN1267, *ECP5 sysDSP Usage Guide*
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1182, *LatticeECP3-DSP sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*.
- DSP (Digital Signal Processing) Technology: sysDSP web page on the Lattice Web site.
- The MathWorks Web site

See Also ▶"Hardware How-To" on page 1162

How to Use the Global Set/Reset (GSR) Signal

This topic provides guidelines and specific instructions for using the Global Set/Reset Interface (GSR) signal of a Lattice FPGA device simulation model for use with all Lattice FPGAs.

Details regarding the architecture of global signals are described in the data sheet for each device family on the Lattice Web site. For more information on the GSR library element, see "GSR" on page 2042.

The items presented in this topic are as follows:

- GSR Hardware Resource
- GSR Usage Cases
  - Inferred GSR
  - User Specified Inferred GSR
  - Global GSR
  - LSR (No GSR)
- Using the GSR Resource with Lattice IPs
- RTL Functional Simulation and the GSR Resource
- Mixed Language Simulation and the GSR Resource
- Using Attributes to Control GSR Resource Usage

GSR Hardware Resource Lattice FPGAs contain both GSR (Global Set Reset) and PUR (Power Up Reset) resources. The GSR hardware resource in Lattice FPGAs provides a convenient mechanism to allow design components to be reset without using any routing resources.

Note

The PUR resource is used to reset the device after configuration is complete when the device is powered on. A PUR component is provided to allow simulation testbenches to simulate this pulse, but this component is never used as part of the design. For more information on PUR, see the How to Use the Power Up Set/Reset (PUR) Global Signal topic.

There are two primary ways to take advantage of the GSR hardware resource in your design: *use the GSR to reset all components on your FPGA* or to *use the GSR to eliminate any routing resources needed for one reset in a multiple reset design*. If there is only one reset signal for the entire design, you would want to use the GSR in the first way, to reset all components on the FPGA. When using the GSR to eliminate any routing resources needed for one reset in a multiple reset design, typically the GSR would be used for the reset with the highest fan-out.

Note

The GSR should only be used for asynchronous active low resets since the hardware will always assert GSR asynchronously. The software will take this into account automatically and will not connect a synchronous reset to GSR when the Inferred GSR or User Specified Inferred GSR modes are used, However there are ways to override the software default behavior. You are responsible for ensuring correct functionality when overriding the normal software defaults or when using the Global GSR usage mode. The following sections of this document detail how to take advantage of this resource with the Lattice Diamond design software.

GSR Usage Cases In Diamond, there are four usage cases with respect to initialization set/resets:

- *Inferred GSR* – In this case the software automatically determines which reset signal has the highest fan-out (for either single or multiple reset designs) and uses the GSR resource as the routing for that reset signal. This usage case is the default condition in Diamond if there is no user-instantiated GSR component in the design.

  This usage case is the best choice for most applications. The software determines the reset with the most loads and uses the GSR resource for that signal which provides the largest reduction in needed routing resources. The Inferred GSR usage case can also be used whether the design has a single or multiple resets.
- *User-Specified Inferred GSR* – This is the same as the Inferred GSR usage except that the reset signal that is specified in the preference (.lpf) file determines which signal uses the GSR resource regardless of the fan-out of the signal.
- *Global GSR* – This case treats the GSR resource as a reset for all elements in the design.
- *LSR (No GSR)* – LSR (local set/reset) specifies that no GSR is to be used, that is, that all resets will use local routing resources instead of using the GSR resource.

Note

In the Inferred GSR and User-Specified Inferred GSR usage cases, the software will only connect elements with an asynchronous reset to GSR. Elements requiring a synchronous reset will use only local routing.

Inferred GSR The Inferred GSR usage case is the simplest to use. If everything is left to default software settings and no GSR component is instantiated in the design, then the software will implement a design using GSR for the reset signal with the highest fan-out.

Inferred GSR is the recommended usage case unless any of the following conditions exist:

- If you need to use a specific reset signal for GSR.
- If you need to globally use GSR even when there are multiple resets.
- If you need to completely disable GSR.

To use the Inferred GSR usage case, there are no design changes necessary. You simply implement the design using Diamond with default settings. The necessary software settings are:

- Synplify Synthesis Properties: **Force GSR: False (default)**
- LSE Synthesis Properties: **Force GSR: Auto (default)**
- Precision Synthesis Properties: **Force GSR: False (default)**
- Map Properties: **Infer GSR: True (default)**

- Preferences: **Do not specify GSR_NET in the .lpf file**

Note

When LSE is enabled, Diamond software infers GSR at the synthesis stage of the process flow. When other synthesis tools are selected, Map infers GSR.

LSE also has the ability to look into black box modules (.ngo files) for any GSR settings. If LSE finds a GSR instance or attribute in a module, LSE will not infer GSR and will instead pass the GSR information to Map to handle.

User Specified Inferred GSR The User Specified Inferred GSR usage case is identical to the Inferred GSR usage case except that the reset signal on which GSR is inferred is specified in the preference (.lpf) file instead of being automatically determined by fan-out.

If default software settings are enabled and no GSR component is instantiated in the design, and a preference exists for GSR_NET, then the software will implement a design using GSR for the reset signal specified by GSR_NET. This usage case is best for a design with multiple resets where a specific reset is required to use GSR regardless of the fan-out of the net.

To use the User-Specified Inferred GSR usage case, there are no design changes necessary. You simply implement the design using Diamond with default settings and a GSR_NET preference specified. The necessary software settings are:

- Synplify Synthesis Properties: **Force GSR: False (default)**
- LSE Synthesis Properties: **Force GSR: False**
- Precision Synthesis Properties: **Force GSR: False (default)**
- Map Design option (Strategies dialog): **Infer GSR: True (default)**
- Preferences: **Specify GSR_NET in the .lpf file.** The syntax of the preference is GSR_NET net “reset_signal_name”.

Global GSR The Global GSR usage case is intended for all elements in a design to be reset using the GSR resource. This usage is a good fit for a design with a single reset. It can also be used with multiple resets in the design, but it can produce unexpected functionality in this case. For example, if the GSR resource is used for the reset with the largest fan-out, elements on a second reset signal will still be reset by the first reset signal. Additionally, any registers with a synchronous reset will be attached to their local reset as well as the GSR reset (which asserts asynchronously).

Note also that the GSR resource is active low. In the case of a register using an active high reset, that register will remain connected to the signal but will have GSR enabled. This will cause the register to remain in reset. If a mix of active low and active high resets are used in a design or a mix of synchronous and asynchronous resets are used, then the Inferred GSR usage case should be used. In the Inferred GSR usage case, the software will automatically take into account whether a reset is synchronous, asynchronous, active high, or active low. In the Global GSR usage case, the software assumes the user knows what the design intent is and makes minimal design changes in order

to allow the user intent to be implemented. Global GSR is meant for a true global reset application.

To use the Global GSR usage case, a GSR component must be instantiated in the design and connected to the signal that is targeted as the reset signal, usually a primary input. If a GSR component is not instantiated in the design, the software will not treat the design as a Global GSR usage case. The GSR component must be instantiated into the design itself, not into the testbench for the design. Below are examples of how to instantiate the GSR library element in both Verilog and VHDL.

GSR VHDL Example:

```
GSR_INST:   GSR port map (GSR=><global reset sig>);
```

GSR Verilog HDL Example:

```
GSR     GSR_INST (.GSR (<global reset sig>));
```

Note that in Verilog the GSR instantiation must be in the top-level module of the design and it must be named GSR_INST. The necessary software settings are:

- Synplify Synthesis Properties: **Force GSR: False (default)**
- LSE Synthesis Properties: **Force GSR: Auto (default)**
- Precision Synthesis Properties: **Force GSR: False (default)**
- Map Design option (Strategies dialog): **Infer GSR: True (default)**. If a GSR component is already instantiated, this property will be ignored.
- Preferences: **Do not specify GSR_NET in the .lpf file**. If a GSR_NET preference is specified and a GSR component is already instantiated, the preference will be ignored.

LSR (No GSR) The LSR (local set/reset) usage case always uses local routing for the reset signals and does not use the GSR resource. This is the recommended usage case if there is a requirement to do timing analysis on the reset signals or if synchronous reset is being used throughout the design.

To use the LSR usage case there must be no GSR instantiated in the design, no GSR_NET preference specified, and the software settings used do not infer any GSR resource. The necessary software settings are

- Synplify Synthesis Properties: **Force GSR: False (default)**
- LSE Synthesis Properties: **Force GSR: False**
- Precision Synthesis Properties: **Force GSR: False (default)**
- Map Design option (Strategies dialog): **Infer GSR: False**
- Preferences: **Do not specify GSR_NET in the .lpf file**.

Using the GSR Resource with Lattice IPs Lattice IPs use the GSR resource when running in evaluation mode and may require it on an individual basis for performance reasons. For all current IP, no individual GSR option is

available. These IPs will function correctly in all usage cases for GSR. For any IP that requires the use of GSR on it's reset for performance issues, the User-Specified Inferred GSR usage case or Global GSR usage case should be used. For the User-Specified Inferred GSR usage case, the reset signal the IP is connected to should be entered in the preferences as the GSR_NET preference.

RTL Functional Simulation and the GSR Resource If the GSR resource is used as a routing replacement for one of the reset signals in a design using the Inferred GSR or User-Specified GSR usage cases, then the functional simulation will be correct for both the RTL (pre-synthesis) design and the netlist (post-synthesis and post-map) versions of the design.

If the GSR resource is used as a global reset, however, the RTL functional simulation will not correctly simulate the reset functionality if the design uses multiple resets. Global GSR causes all Lattice components in the design to respond to the reset used for the GSR whether they are connected to that reset or not. In the RTL representation the use of the GSR resource is not modeled in synthesizable RTL code, but in the post-map netlist, the entire design will respond correctly to the reset used for GSR. This is because the design will now consist of Lattice components that have been modeled to take the GSR information into account as part of their functionality. The post-map design will reset all elements sensitive to GSR when the reset signal assigned to GSR is used. Even if an element is connected to a different reset, it will still reset when the GSR signal is used if it is sensitive to GSR.

If the design contains both RTL code and Lattice component instantiations, the design may need some additions to prevent simulation problems. Verilog simulations will produce errors if there are library elements in the design that use the GSR information and instantiations of both GSR and PUR components are not present in the top-level of the design. For Verilog designs, you must instantiate the GSR component in the top level of the design hierarchy with the instance name of GSR_INST if the design contains any instantiation of library elements that are affected by the GSR settings (sequential and memory components in general). Additionally, you must also instantiate the PUR component in the top level of the design hierarchy with the instance name of PUR_INST for these same components. Below are examples of the Verilog instantiations of GSR and PUR.

GSR Verilog HDL Example:

```
GSR     GSR_INST (.GSR (<global reset sig>));
```

PUR Verilog HDL Example:

```
PUR     PUR_INST (.PUR (<powerup reset sig>));
```

If these instantiations are not present, you will see errors in your simulator. Following is an example error produced in Aldec Active-HDL if the GSR instantiation is not present.

```
ELBREAD: Error: Hierarchical reference not found:
GSR_INST.GSRNET (from "tb.__.EFBInst_0")
KERNEL: Error: E8005 : Kernel process initialization failed.
```

```
VSIM: Fatal error: Fatal error occurred during simulation
initialization.
```

If the PUR instantiation is not present a similar error will be shown for PUR_INST.PURNET.

If any usage case other than the Global GSR usage case is desired, the GSR and PUR components must be removed from the top-level hierarchy before synthesizing the design (Translate Database process step) if the top level is part of the design. The one exception to this requirement is if the GSR and PUR components are connected to VCC instead of a design signal. The GSR and PUR components do not require removal if they are connected to VCC. If the top level of the design hierarchy is the test bench, then the GSR and PUR components do not need to be removed for any of the usage cases.

Mixed Language Simulation and the GSR Resource The ability to simulate the GSR resource is handled differently in Verilog and VHDL due to differences in the languages. This difference can make simulating the GSR functionality in a mixed language difficult.

Verilog simulation models access the GSR state by referring to a signal inside the GSR component that must be instantiated at the top level of the design. This can be seen in the following piece of Verilog code inside a simulation library module:

```
parameter GSR = "ENABLED";

tri1 GSR_sig = GSR_INST.GSRNET;
tri1 PUR_sig = PUR_INST.PURNET;

always @ (GSR_sig or PUR_sig ) begin
if (GSR == "ENABLED") begin
SRN = GSR_sig & PUR_sig ;
end
else if (GSR == "DISABLED")
SRN = PUR_sig;
end
```

VHDL simulation models access the GSR state by referring to a signal in a package that is used by the model. Below are sections of the package and its use in a simulation model.

```
-------------Package Declaration----------------------
PACKAGE global IS
SIGNAL gsrnet: std_logic := 'H';
SIGNAL purnet: std_logic := 'H';
SIGNAL tsallnet: std_logic := 'H';
END global;

------------Entity Usage Statement------------------
USE work.global.gsrnet;
USE work.global.purnet;

-------------Entity Declaration-------------------------
GENERIC (
```

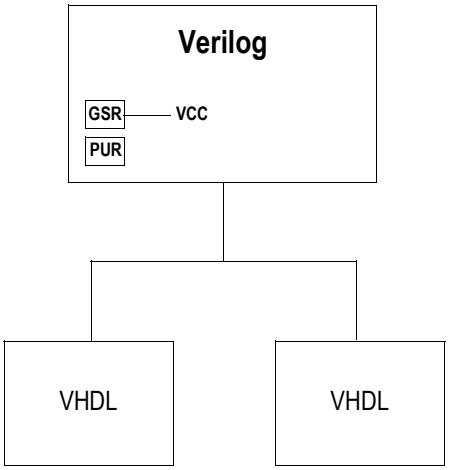
```
gsr             : String  := "ENABLED");

-------------Architecture Body------------------------
IF (gsr = "DISABLED") THEN
set_reset := purnet;
ELSE
set_reset := purnet AND gsrnet;
END IF;
```

If the desired simulation is post-synthesis, or post-map, the easiest solution is to generate a netlist in the same language as the top-level testbench. Then the simulation can be done in a single language with respect to GSR and the correct functionality can be simulated. This occurs automatically if the simulation is run from Diamond by selecting the testbench and then selecting **Post-Route Functional Simulation** or **Post-Route Timing Simulation**.

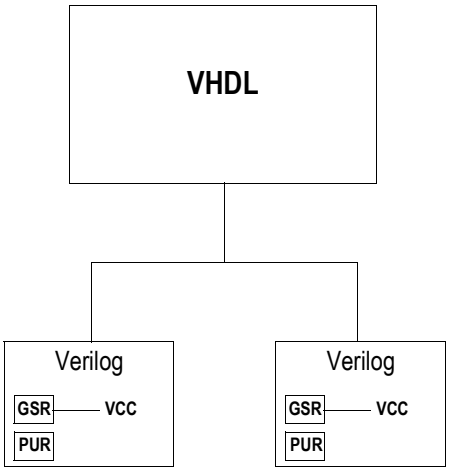
If the simulation is an RTL functional simulation, the same issues described above in the RTL functional simulation section exist along with the complication of the differences in how the languages deal with GSR. There are two different issues (initialization and functionality) and different usage cases to be considered in mixed language simulation. The different relevant combinations are discussed below.

- *Simulation Initialization, Verilog top, VHDL bottom*. When a simulator starts or initializes a design, it elaborates the design and assembles all the required information. In Lattice VHDL simulation libraries, components access GSR information via a VHDL package.



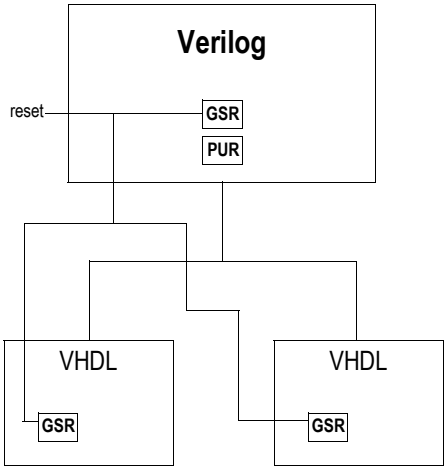
The package is provided with the simulation library so no extra information is required to initialize the simulator. In Lattice Verilog simulation libraries, the GSR information is accessed from an instantiation at the top of the Verilog part of the design. This requires a GSR and PUR component instantiation as shown in the section, RTL Functional Simulation and the GSR Resource. These instantiations are required in the Verilog hierarchy if it contains any Lattice Verilog library components that access GSR information even if GSR is not functionally used in the design. For this design structure, the Verilog GSR and PUR instantiation must be at the top of the hierarchy, either the top-level netlist or the test bench.

- *Simulation Initialization, VHDL top, Verilog bottom*. When a simulator starts or initializes a design, it elaborates the design and assembles all the required information. In Lattice VHDL simulation libraries, components access GSR information via a VHDL package.



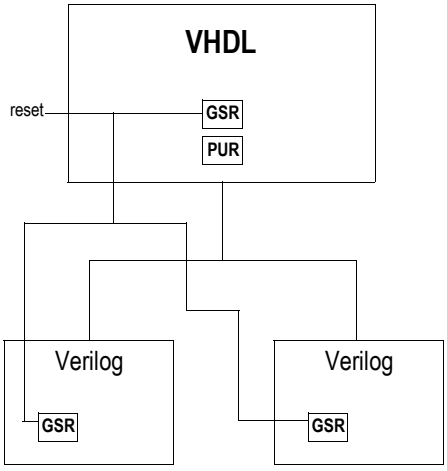
The package is provided with the simulation library so no extra information is required to initialize the simulator. In Lattice Verilog simulation libraries, the GSR information is accessed from an instantiation at the top of the Verilog part of the design. This requires a GSR and PUR component instantiation as shown in the section, RTL Functional Simulation and the GSR Resource. These instantiations are required in the Verilog hierarchy by any Lattice Verilog library component that accesses GSR information even if GSR is not functionally used in the design. For this design structure, the Verilog GSR and PUR instantiations must be at the top of the Verilog hierarchy, not the top of the whole hierarchy since the top is VHDL. If the design has a VHDL top and there are multiple Verilog instantiations or modules within the VHDL design, then each separate Verilog hierarchy must have its own GSR and PUR component instantiation. The GSR and PUR component instantiations in one Verilog hierarchy are not visible to other separate hierarchies.

- *Global GSR functional simulation, Verilog top, VHDL bottom*. Since the GSR information for Lattice simulation libraries is simulated in Verilog and VHDL using different mechanisms, it is required to have a GSR component instantiation in each language section.



A GSR component should be instantiated in the Verilog top hierarchy and connected to the desired reset signal. A GSR component should be instantiated in the VHDL hierarchy and connected to the same reset signal. If there are separate VHDL hierarchies within the Verilog top hierarchy, each VHDL hierarchy must have its own GSR component instantiated and connected to the reset signal. GSR components instantiated in one VHDL hierarchy do not affect other VHDL hierarchies. A PUR component is still required in the Verilog for simulation initialization purposes.

- *Global GSR functional simulation, VHDL top, Verilog bottom*. Since the GSR information for Lattice simulation libraries is simulated in Verilog and VHDL using different mechanisms, it is required to have a GSR component instantiation in each language section.



A GSR component should be instantiated in the VHDL top hierarchy and connected to the desired reset signal. A GSR component should be instantiated in the Verilog hierarchy and connected to the same reset signal. If there are separate Verilog hierarchies within the VHDL top hierarchy, each Verilog hierarchy must have its own GSR component instantiated and connected to the reset signal. GSR components instantiated in one Verilog hierarchy do not affect other Verilog hierarchies.

- *Inferred GSR, User Specified Inferred GSR, and LSR (No GSR) functional simulation*. Unlike the Global GSR usage case, the functionality of the design can be simulated correctly for the other usage cases without requiring special simulation techniques since the functionality will be simulated using the reset signals in the design. However, the issue with initializing the Verilog section of the design is the same as described in the simulation initialization sections above. A GSR and PUR component instantiation must be present in the Verilog hierarchy or in each separate Verilog hierarchy if there are multiple ones under a VHDL top hierarchy. The signal input to the GSR component can be assigned to the desired reset signal or to a logical value of '1' if the GSR functionality is not used.

Using Attributes to Control GSR Resource Usage This section contains details on how to control the behavior of GSR usage on a module or component level. This information is intended for expert usage only. This information is not required for any of the usage flows discussed earlier in this topic to function correctly.

One example of the need to use this information is in the case of using Global GSR. If one section of the design is required to continue functioning during the use of reset (such as a communication port), then it is possible to use attributes to prevent the GSR resource from being implemented in a particular hierarchy or component. Likewise, it is possible to force a hierarchy or component to respond to GSR even if it was on a different reset in the Inferred GSR case. To correctly use these attributes, it is necessary to understand both the hierarchy inheritance order and how each attribute is treated in the different usage cases.

GSR Attribute Values and Syntax You must specify which portions of the design that you wish to alter the way in which they respond to the GSR reset signal. Unless specified otherwise, by default, all design elements will respond to the global reset signal if it is present. The available values are as follows.

- ENABLED – This is the default value on most library elements. This value allows the software to determine the final value and will be overridden by the parent hierarchy if the parent has a value of anything other than ENABLED.
- DISABLED – This prevents the hierarchy or element from responding to the GSR value. It cannot be changed by the parent's value.
- FORCEENABLE – This forces the hierarchy or element to respond to the GSR value. It cannot be changed by the parent's value.
- IPENABLE – This forces the hierarchy or element to respond to the GSR value when IP is being used in evaluation mode. It cannot be changed by the parent's value. This is only meant for internal Lattice IP to use. This value should never be used within a design itself.

These values are then placed as synthesis attributes in the design which synthesis will pass to the Map program. The attributes can be placed on any instance whether it is a level of hierarchy or a leaf component. Map will then interpret the attributes along with the usage case and make the necessary changes to the design. After Map is finished, all elements in the design will have GSR values of either ENABLED or DISABLED only. The syntax of the

attributes is shown below for Verilog and VHDL with “VALUE” shown as the placeholder for legal values.

Synplify or LSE GSR Attributes Verilog GSR attribute syntax:

```
modname modinst (signal list) /* synthesis GSR=VALUE */;
```

VHDL GSR attribute syntax:

```
attribute GSR : string;
attribute GSR of comp_instance: label is "VALUE";
component_instance: component_name port map (signal list);
```

Note

Replace the placeholder VALUE in the above syntax examples with legal values from the value list.

Precision GSR Attributes Verilog GSR attribute syntax:

```
modname modinst (signal list) //exemplar attribute modinst GSR
VALUE;
```

VHDL GSR attribute syntax:

```
attribute GSR : string;
attribute GSR of comp_instance: label is "VALUE";
component_instance: component_name port map (signal list);
```

Note

Replace the placeholder VALUE in the above syntax examples with legal values from the value list.

Determining Actual Pre-Map GSR Attribute Value Following are the precedence rules to use to determine the “actual” GSR value an instance has in the pre-map netlist.

The first column in the table below represents the possible explicit values an instance can have. An explicit value is one which has been put directly on the instance. The first row represents the possible implicit values a library element can have via GSR attribute set at a parent module that includes the

instance. The “parent” term is used loosely here as more precisely, the GSR value could be sitting on one or more levels of hierarchy above the parent.

Table 137: Determining “Actual” GSR Value

Explicit child GSR value in pre-map netlist	Parent module GSR value (explicit)				
	<none>	DISABLED	ENABLED	FORCEENABLE	IPENABLE
<not set> (component default value ENABLED)	ENABLED	DISABLED	ENABLED	FORCEENABLE	IPENABLE
<not set> (component default value DISABLED)	DISABLED	DISABLED	DISABLED	DISABLED	DISABLED
<not set> (hierarchy level not leaf component)	ENABLED	DISABLED	ENABLED	FORCEENABLE	IPENABLE
DISABLED	DISABLED	DISABLED	DISABLED	DISABLED	DISABLED
ENABLED	ENABLED	DISABLED	ENABLED	FORCEENABLE	IPENABLE
FORCEENABLE	FORCEENABLE	FORCEENABLE	FORCEENABLE	FORCEENABLE	FORCEENABLE
IPENABLE	IPENABLE	IPENABLE	IPENABLE	IPENABLE	IPENABLE

Determining Final Post-Map GSR Attribute Value Following are the rules that the map software uses to determine the GSR attribute value of the physical component that a pre-map library element gets mapped into in the post-map netlist.

The first column represents the possible “actual” values that a pre-map library element can have. The top row represents the usage case map is operating under.

The final post-map GSR value a component has will depend on the following conditions.

- The usage case (Global GSR or either Inferred or User Specified Inferred)
- If the component is on the inferred reset domain or not
- If any Lattice IP is being used in evaluation mode

If the “actual” value of a pre-map library element is,

- DISABLED - It will be mapped to a component which has GSR=DISABLED regardless of the usage case.
- ENABLED – This value allows the software to automatically determine the necessary value, so final value depends on the usage case
  - Global GSR - Final value is ENABLED since this should respond to the global reset

- Inferred GSR or User Specified Inferred GSR & element is on the inferred reset - Final value is ENABLED since this will cause the element to respond to the reset being put on to the GSR resource
- Inferred GSR or User Specified Inferred GSR & element is NOT on the inferred - Final value is DISABLED since this component is not on the inferred reset and therefore should not respond to GSR resource

- FORCEENABLE – This will always be ENABLED regardless of the usage case. This is how a component can be reset by GSR resource in the Inferred GSR usage case even if that component is not on the inferred reset.
- IPENABLE – This will always be ENABLED regardless of the usage case except when an IP license is present and the IP is not on the inferred reset. In this case only, GSR will be set to DISABLED.

Note

A consequence of the IPENABLE functionality is that the IP reset will function differently when it is not on the inferred GSR reset between licensed and evaluation mode. This is an unavoidable consequence of how the evaluation mode functions. It is important to note the licensed mode is functionality correct and will not cause any functional issues in the final design. This issue is not present in the Global GSR usage case.

Table 138: Determining Post-Map GSR Value

Actual Pre-Map Value	**Global GSR Usage Case**	**Inferred GSR and User-Specified Inferred GSR Usage Cases**		
		On Inferred Reset	No Inferred Reset	No Inferred Reset plus IP Evaluation Mode
DISABLED	DISABLED	DISABLED	DISABLED	DISABLED
ENABLED	ENABLED	ENABLED	DISABLED	DISABLED
FORCEENABLE	ENABLED	ENABLED	ENABLED	ENABLED
IPENABLE	ENABLED	ENABLED	DISABLED	ENABLED

Issues Using Global GSR and GSR=DISABLED Attribute Due to Synthesis Optimizations In the GSR flow synthesis does not perform any GSR inferencing done in legacy Lattice software releases. Additionally, synthesis does not understand the attributes used for GSR control, but will pass them in the design to the map software which will perform the correct changes.

However, synthesis normally will perform optimizations it is aware of. Synthesis will see a GSR component that has been instantiated in a design for the Global GSR usage case and may automatically optimize out the signal connections for the reset signal to components if an active low reset signal is defined. However, since synthesis is unaware of the meaning of the GSR attributes, any components in a Global GSR usage case that have GSR=DISABLED on them may be left without any ability to be reset (reset

signal disconnected and GSR=DISABLED). Note that this issue will not occur on the Inferred GSR or User Specified Inferred GSR usage cases.

In order to avoid this issue the User Specified Inferred GSR usage case can be used. The desired reset signal should be specified in the preferences as defined for this usage case. Additionally at the top of the design, the attribute GSR=FORCEENABLED should be specified. This will force all components, even in a multiple reset design, to be reset by GSR similar to the Global GSR usage case. Design elements that need to be prevented from resetting can use the GSR=DISABLED attribute to block reset for those components or hierarchies since they will not be affected by the GSR=FORCEENABLED as shown in Table 137.

Issues with Synthesis Hierarchy Flattening or Optimization Synthesis tools can potentially flatten a hierarchy or optimize hierarchies to improve results. When this occurs, attributes assigned to a hierarchy level will be lost. This is true for the GSR attributes since synthesis is not aware of any special meaning for these attributes. If this occurs, you can prevent the flattening from occurring by adding an additional attribute to the hierarchy.

Below are code examples that illustrate how to preserve hierarchical attributes for the Synplify and Precision synthesis tools:

Synplify or LSE Hierarchy Attributes Verilog hierarchy attribute syntax:

```
modname modinst (signal list) /* synthesis syn_hier=hard */;
```

VHDL hierarchy attribute syntax:

```
attribute syn_hier : string;
attribute syn_hier of comp_instance: label is "hard";
component_instance: component_name port map (signal list);
```

Precision Hierarchy Attributes Verilog hierarchy attribute syntax:

```
modname modinst (signal list) //pragma attribute modinst
hierarchy preserve;
```

VHDL hierarchy attribute syntax:

```
attribute syn_hier : string;
attribute syn_hier of comp_instance: label is "preserve";
component_instance: component_name port map (signal list);
```

See Also ▶"Hardware How-To" on page 1162

How to Use the Power Up Set/Reset (PUR) Global Signal

This topic provides guidelines and specific instructions for interfacing to the Power Up Set/Reset (PUR) signals of a Lattice FPGA device simulation model for use with all Lattice FPGAs. Details regarding the architecture of global signals are described in the data sheet for each device family. For more information on the PUR library element, see "PUR" on page 2375. For more information on using Lattice library elements in HDL, see Lattice Synthesis Header Libraries topic.

About the GSR and PUR Signals Both Global Set/Reset (GSR) and PUR are global RESET signals that will initialize the chip to some known reset state. In a given device, GSR has a dedicated input pad; however, PUR is not driven by any external pad. PUR is only activated upon device startup, just after configuration is complete, which is basically power-up of the device. Unlike PUR, the GSR signal can be called at any time during operation to reset the devices initial values.

Major items discussed in this topic are as follows:

- Using the PUR Library Element
- PUR Verilog HDL Example
- PUR VHDL Example

Using the PUR Library Element The Power Up Set/Reset (PUR) signal is activated during device configuration and is driven by internal circuitry. PUR is often used in the context of a system-level simulation where the power up control of the PC board is verified. You can model the behavior of this signal by adding the PUR library element to your test fixture. When the PUR input port is driven all register elements of the design will respond as if a global set/reset was asserted. PUR is in an active LOW state by default.

You commonly instantiate the PUR element in the test fixture with the instance name PUR_INST.

Note: Verilog simulations will produce errors if there are library elements in your design that require the instantiation of GSR, PUR, and TSALL elements and they are not present. For Verilog designs, you must instantiate the PUR and GSR elements in their top-level netlists with instance names PUR_INST and GSR_INST respectively if the design contains any instantiation of library elements which are affected by GSR.

The examples below shows proper syntax for instantiating a PUR element in Verilog HDL or VHDL.

PUR Verilog HDL Examples

```
PUR     PUR_INST (.PUR (<powerup reset sig>));
```

The PUR element instantiation below also shows a parameter called RST_PULSE with a value of 10 ns that is used to set the required reset pulse

length. You can pass a required numerical value such as 10 ns or 100 ns. If you do not specify this parameter, by default it will be set at 1 ns.

```
PUR     PUR_INST (.PUR (<powerup reset sig>));
defparam PUR_INST.RST_PULSE = 10;
```

PUR VHDL Examples `PUR_INST: PUR port map (PUR=><powerup reset sig>);`

As shown in the Verilog PUR example, this VHDL PUR element instantiation below also shows a parameter called RST_PULSE with a value of 10 ns that is used to set the required reset pulse length.

```
PUR_INST : PUR
generic map (RST_PULSE => 10)
port map (PUR => <signal>);
```

Note: PUR library elements should be used as part of a Verilog test fixture or VHDL test bench to model power up set/reset only. PUR library elements can not interface to a signal of the FPGA design itself.

References For more information refer to the following documentation.

- PUR library element description in "PUR" on page 2375.

See Also "Hardware How-To" on page 1162

How to Use the Tristate Interface (TSALL) Global Signal

This topic provides guidelines and specific instructions for interfacing to the Global Tristate Interface (TSALL) signal of a Lattice FPGA device for use with the Lattice MachXO and LatticeSC/M FPGA devices. Details regarding the architecture of global signals are described in the data sheet for each device family. For more information on the TSALL library element, see "TSALL" on page 2428. For more information on using Lattice library elements in HDL, see Lattice Synthesis Header Libraries topic.

Major items discussed in this topic are as follows:

- Using the TSALL Library Element
- TSALL Verilog HDL Example
- TSALL VHDL Example

Using the TSALL Library Element By default the tristate enable of PIO blocks will be assigned to local output enable signals of the design. To explicitly connect a design signal to the global tristate network of the device then it is necessary to instantiate the TSALL library element in your design. TSALL is considered active LOW to enable output.

Important

You must instantiate the TSALL with the instance name TSALL_INST. The port name for the MachXO TSALL library element is TSALL. The port name for the LatticeSC/M TSALL library element is TSALLN.

Note

Verilog simulations will produce errors if there are library elements in your design that require the instantiation of GSR, PUR, or TSALL elements and they are not present.

The examples below show proper syntax for instantiating the MachXO and LatticeSC/M TSALL element in Verilog HDL or VHDL, respectively.

TSALL Verilog HDL Example (MachXO)

```
TSALL   TSALL_INST (.TSALL ());
```

TSALL VHDL Example (MachXO)

```
component TSALL
        port( TSALL: in STD_ULOGIC );
    end component;
    -- Attributes for Synplify
    attribute syn_black_box: boolean ;
    attribute syn_black_box of TSALL: component is true;
    attribute syn_noprune: boolean ;
    attribute syn_noprune of TSALL: component is true;
    -- Attributes for Precision RTL
    attribute BLACK_BOX : boolean;
    attribute BLACK_BOX of TSALL: component is true;
    attribute DONT_TOUCH : boolean;
    attribute DONT_TOUCH of TSALL_INST: label is true;
begin

    TSALL_INST: TSALL port map (TSALL=><global tristate sig>);
```

TSALL Verilog HDL Example (LatticeSC/M)

```
TSALL   TSALL_INST (.TSALLN ());
```

TSALL VHDL Example (LatticeSC/M)

```
component TSALL
        port( TSALLN: in STD_ULOGIC );
    end component;
    -- Attributes for Synplify
    attribute syn_black_box: boolean ;
    attribute syn_black_box of TSALL: component is true;
    attribute syn_noprune: boolean ;
    attribute syn_noprune of TSALL: component is true;
    -- Attributes for Precision RTL
    attribute BLACK_BOX : boolean;
    attribute BLACK_BOX of TSALL: component is true;
    attribute DONT_TOUCH : boolean;
    attribute DONT_TOUCH of TSALL_INST: label is true;
```

```
begin
    TSALL_INST: TSALL port map (TSALLN=><global tristate sig>);
```

Note

Lattice Applications recommends using “don't touch” synthesis directives, like syn_noprune and DONT_TOUCH, with library elements that you instantiate in VHDL designs. For more information on Lattice library elements, see “FPGA Libraries Reference Guide” on page 1665.

References For more information refer to the following documentation.

- TSALL library element description in “TSALL” on page 2428

See Also “Hardware How-To” on page 1162

How to Use the Internal Oscillator

This topic provides guidelines and specific instructions for instantiating the internal oscillator (OSCA, OSCC, OSCD, OSCE, OSCF, OSCG, OSCH, OSCI, OSCJ) element in your Verilog and VHDL code for a given device family. It also contains specific code examples. For more information on these library elements, see “FPGA Libraries Reference Guide” on page 1665. For more information on using Lattice library elements in HDL, see Lattice Synthesis Header Libraries topic.

Major items discussed in this topic are as follows:

- Using Internal Oscillator Library Elements
- Internal Oscillator Verilog HDL Example
- Internal Oscillator VHDL Example

Using Internal Oscillator Library Elements To interface a user-defined clock signal to the internal oscillator, instantiate the appropriate library element from “FPGA Libraries Reference Guide” on page 1665. The following library elements are internal oscillators for their respective architectures:

- OSCA (LatticeSC/M)
- OSCC (MachXO)
- OSCD (LatticeECP2/M)
- OSCF (LatticeECP3)
- OSCE (LatticeXP2)
- OSCG (ECP5)
- OSCH (MachXO2)
- OSCI (LIFMD)
- OSCJ (MachXO3D)

For information on internal clock frequencies, clock divider settings, default settings, and other details, see "FPGA Libraries Reference Guide" on page 1665.

Note: Lattice Applications recommends using "don't touch" synthesis directives with library elements that you instantiate in your design.

The examples below shows proper syntax for instantiating an internal oscillator library element in Verilog HDL or VHDL for miscellaneous internal oscillators.

OSCD Verilog Example

```
module OSC_TOP (OSC_CLK);
output   OSC_CLK;
OSCD OSCinst0 (.CFGCLK(OSC_CLK));
defparam OSCinst0.NOM_FREQ = "2.5" ;
endmodule
```

OSCE VHDL Example

```
COMPONENT OSCE
-- synthesis translate_off
GENERIC (NOM_FREQ: string := "2.5");
-- synthesis translate_on
PORT (OSC :OUT std_logic);
END COMPONENT;

attribute NOM_FREQ : string;
attribute NOM_FREQ of OSCinst0 : label is "2.5";

begin

OSCInst0: OSCE
-- synthesis translate_off
GENERIC MAP (
NOM_FREQ => "2.5"
)
-- synthesis translate_on
PORT MAP (
OSC => osc_int
};
```

References For more information on using Lattice library elements in HDL refer to the following documentation.

- OSCA library element description (FPGA Libraries Reference Guide)
- OSCC library element description (FPGA Libraries Reference Guide)
- OSCD library element description (FPGA Libraries Reference Guide)
- OSCE library element description (FPGA Libraries Reference Guide)
- OSCF library element description (FPGA Libraries Reference Guide)

See Also "Hardware How-To" on page 1162

Chapter 17

Constraints Reference Guide

Constraints are instructions that you specify to guide your design toward desired results and performance goals. In the Lattice Diamond design flow, constraints include logical preferences and HDL attributes.

You can assign constraints using one or both of the following methods:

- Assign post-synthesis design preferences using the logical preference file (.lpf).

  See the "Preferences" on page 1195 section for complete descriptions of each preference, including syntax rules and examples.

- Assign HDL or schematic-based attributes using the design source files. These attributes are used to direct Map and Place & Route.

  See the "HDL Attributes" on page 1282 section for complete descriptions of each attribute, including conventions and examples.

If you are using the Lattice Synthesis Engine (LSE), constraints will also include Synopsys Design Constraints (SDC) as well as HDL attributes and directives that influence the optimization or structure of the output netlist. See the "Lattice Synthesis Engine (LSE) Constraints" on page 1339 section for descriptions of these LSE constraints.

See Also "Applying Design Constraints" on page 360

- "Using Preference Scripts" on page 536
- "Integrated Synthesis" on page 583
- "Applying Design Constraints" on page 360

Preferences

Preferences are implementation constraints for assigning design logic to physical resources. Preferences, unlike HDL attributes, are written to the logical preference file (.lpf).

The following table shows the device support for each preference and the preferences that are supported by TRACE.

Preference	Devices	TRACE
ASIC	all	✓
BANK	all	
BLOCK	all	✓
CLKDELAY	LatticeSC/M only	✓
CLKSKEWDIFF	all	✓
CLKSKEWDISABLE	all	✓
CLOCK_TO_OUT	all	✓
CUSTOM_IDCODE	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	
DEFINE BUS	all	✓
DEFINE GROUP	all	✓
FBEXTDELAY PCM	all	✓
FREQUENCY	all	✓
GSR_NET	all	
HGROUP	all	✓
INPUT_SETUP	all	✓
IOBUF	all	
LOCATE	all	
LOCATE VREF	all	
MAXDELAY	all	✓
MAXSKEW	all	✓
MULTICYCLE	all	✓
OUTPUT LOAD	all	✓
PERIOD	all	✓

Preference	Devices	TRACE
PRIORITIZE	all	
PROHIBIT	all	
PROHIBIT EDGE	LatticeECP2/M	
PROHIBIT PRIMARY	all	
PROHIBIT SECONDARY	all except ECP5	
REGION	all	
RVL_ALIAS **Note**: RVL_ALIAS is automatically generated and should not be added manually or edited.	all	✓
SSO	all	
SYSCONFIG	all	
SYSTEM_JITTER	all	✓
TEMPERATURE	all	✓
TRACEID	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2	✓
UGROUP	all	
UNIQUE_ID	ECP5	
USE DIN CELL	all except MachXO, MachXO2, MachXO3D, MachXO3L, MachXO3D	
USE DOUT CELL	all except MachXO, MachXO2, MachXO3D, MachXO3L, MachXO3D	
USE EDGE	LatticeECP2/M	
USE EDGE2EDGE	LatticeECP3 EA, MachXO2	
USE PRIMARY	all	
USE PRIMARY2EDGE	LatticeECP2/M LatticeECP3 LatticeSC/M	
USE SECONDARY	all except ECP5	

Preference	Devices	TRACE
USERCODE	all	
VCC_DERATE	all	✓
VCC_NOMINAL	all	✓
VCC1P2_DERATE	LatticeSC/M	✓
VCCA_DPHY0_DERATE / VCCA_DPHY1_DERATE	LIFMD/F	✓
VCCAUX_DERATE	all	✓
VCCIO_DERATE	all	✓
VOLTAGE	all	✓

See Also ▶"Preferences and the Design Flow" on page 377

ASIC

Specifies either black box or ASIC cell element programming properties.

Device Support All

Syntax ASIC <*asic_instance_name*> TYPE <*lib_element_type*> (<*keyword*>=<*value*>)+;

where:

<*asic_instance_name*> ::= ASIC instance name (string)

<*lib_element_type*> ::= Library element type (string)

<*keyword*> ::= string

<*value*> ::= string

Example:

```
ASIC "my_dll" TYPE "CIMDLLA" CLKOP_PHASE=90 PHASELOCK=DISABLED;
```

The ASIC instance names and their types in a design are listed in the "ASIC Components" section of the Map Report.

ASIC Types for Implementing the LatticeSC SMI Bus For LatticeSC and LatticeSCM devices, an ASIC TYPE must be specified when assigning a serial management interface offset value (SMI_OFFSET) for PLLs, DLLs, or select MACO blocks. The SMI_OFFSET value must be set in the preference file for mapping, placement and routing.

DLL Example:

```
ASIC “dll/dll_0_0” TYPE “CIMDLLA” SMI_OFFSET=”0x420”;
```

PLL Example:

```
ASIC “pll/pll_0_0” TYPE “EHXPLLA” SMI_OFFSET=“0x410“;
```

MACO Example:

```
ASIC "flxmc_sys_ge/flxmc_top_sys/flxmc_top_ebr/flxmc_top_mib"
TYPE "FLXMC" SMI_OFFSET="0x500";
```

LatticeSC DLL and PLL ASIC Types The following list shows the ASIC TYPE keyword for each LatticeSC PLL and DLL that can be implemented with the SMI system bus.

- CIDDLLA – DLL clock injection delay removal. Reduces clock injection delay, which is the delay from the input pin of the device to a destination element such as a flip-flop.
- CIMDLLA – DLL clock injection delay match. Allows two synchronous clock inputs to be used to create a digital control (DCNTL) vector that is the delta of the delay between the two clocks.
- SDCDLLA – single delay cell dll. Corrects for clock injection and enables the 9-bit ALU output
- TRDLLA – time reference delay. Generates four phases of the clock—0, 90, 180, 270 degrees—along with the control setting used to generate these phases.
- EHXPLLA – enhanced extended performance PLL.

See *Technical Note 1098, LatticeSC sysCLOCK PLL/DLL User’s Guide,* for more information.

LatticeSCM MACO ASIC Types The following list shows the MACO blocks that can be implemented with the SMI system bus and the ASIC type keywords for each.

- FLXMC – ethernet flexiMAC, a media access controller. The FLXMC includes the following ASIC type:
  - FLXMC.

  See *User Guide 48, LatticeSCM Ethernet flexiMAC MACO Core*, for more information.
- SPI4.2 – system packet interface level 4 IP core. The ASIC type for the SPI4.2 depends on the size of the device and the site used for the SPI4.2 MACO. Therefore, the SPI4.2 includes the following ASIC types:
  - SPI4_15LLM0

- SPI4_25LLM0
- SPI4_25RLM0
- SPI4_40LLM0
- SPI4_40RLM0
- SPI4_80LLM0
- SPI4_80RLM0
- SPI4_115LLM0
- SPI4_115RLM0

See *User Guide 44, LatticeSCM SPI4.2 MACO Core,* for more information.

BANK

Sets the VCCIO voltage level of the IO bank. In cases where design signals are not assigned to a device site or bank (floating), the BANK preference can be applied to direct the IO placer algorithm of Place and Route (par) to assign only design IO signals with compatible signal standards to the bank.

Device Support All

Note

For LatticeSC devices, you must use the configuration bank, Bank 1, to set the VCCIO voltage level. See BANK 1, following.

For more details on assigning signal standards to a design signal, see the IOBUF preference. See the LOCATE preference for details on assigning I/O signals to a device site.

Usage The BANK preference can be set manually in the logical preference file. For MachXO2, MachXO3D, and MachXO3L devices, it can also be set in the Global preference sheet of Spreadsheet View.

Syntax BANK <*n*> VCCIO <*x*> V;

where:

<*n*>= bank number

<*x*> can be any compatible voltage supply to that bank; for example, 2.5, 3.3

Care must be taken not to create conflicting preferences that locate I/O signals to a bank that are incompatible with the voltage supply designated by the BANK preference. VCCIO voltage is automatically set when an I/O signal is assigned to a package pin using the LOCATE preference.

LatticeSC/M devices are unique, because the user might need to float I/O signals yet specify the voltage that a bank is running at to enable the 3.3V

protection circuit. Any bank could be attached to a 3.3V power supply on the board, and even if no user I/O signals are assigned to it, this will lead to a hot device and/or excess power consumption.

Example – 3.3V VCCIO If 3.3V is defined, then bank *n* will run as if a 3.3V VCCIO voltage will be supplied:

- PAD Specification File (<*project*>.pad) will report bank *n* as having VCCIO of 3.3V
- PAR will not allow the placement of any I/O signals that are incompatible with 3.3V VCCIO. An error will be generated if there is a LOCATE of an incompatible I/O signal assigned to bank *n*.
- The 3.3V protection circuit will be enabled.

Example – 2.5V VCCIO If a value other than 3.3V is defined—for example, 2.5V—then bank *n* will run as if that VCCIO voltage will be supplied:

- PAD Specification File (<*project*>.pad) will report bank *n* as having VCCIO of 2.5V
- PAR will not allow the placement of any I/O signals that are incompatible with 2.5V VCCIO. An error will be generated if there is a LOCATE of an incompatible I/O signal assigned to Bank *n*.
- The 3.3V protection circuit will be disabled.

Example – Undefined VCCIO Values If no BANK preference is defined, then:

- PAD Specification File (<*project*>.pad) will report bank *n* as having the voltage that is required by the I/O signals placed there. If no design I/O signals are placed into bank *n*, then default 2.5V is used.
- If the VCCIO in the PAD Specification File (<*project*>.pad) is 3.3V, then the 3.3V protection circuit will be enabled; otherwise, it will be turned off.
- It is the user's responsibility to supply the VCCIO voltages reported in the PAD Specification File to the device.

BANK 1 A requirement for LatticeSC devices is that all banks be powered during power-up. Because of this requirement, a configuration bank (Bank 1), must be used to set VCCIO voltage for LatticeSC devices. The BANK 1 preference directs the Place and Route (PAR) engine to place only IO signals with compatible standards into the Bank 1. By ensuring that pins in Bank 1 are not set to a different voltage level than the configuration pins, the BANK 1 preference helps prevent overheating.

BANK 1 Syntax BANK 1 VCCIO <*x*> V;

where:

<*x*> can be any compatible voltage supply to Bank 1; for example, 2.5, 3.3

BANK 1 Example – 3.3V VCCIO If 3.3V is defined, then Bank 1 will run as if a 3.3V VCCIO voltage will be supplied:

- PAD Specification File (<*project*>.pad) will report Bank 1 as having VCCIO of 3.3V
- PAR will not allow the placement of any I/O signals that are incompatible with 3.3V VCCIO. A PAR error will be generated if there is a LOCATE of an incompatible IO to Bank 1.
- The 3.3V protection circuit will be enabled.

BANK 1 Example – 2.5V VCCIO If a value other than 3.3V is defined—for example, 2.5V—then Bank 1 will run as if that VCCIO voltage will be supplied:

- PAD Specification File (<*project*>.pad) will report Bank 1 as having VCCIO of 2.5V
- PAR will not allow the placement of any IOs that are incompatible with 2.5V VCCIO. An error will be generated if there is a LOCATE of an incompatible IO to Bank 1.
- The 3.3V protection circuit will be disabled.

BANK 1 Example – No Bank 1 Preference defined If no BANK 1 preference is defined, then:

- PAD Specification File (<*project*>.pad) will report Bank 1 as having the voltage that is required by the I/O signals placed there. If no design I/O signals are placed into Bank 1, then the default 2.5V is used.
- If the VCCIO in the PAD Specification File is 3.3V, then the 3.3V protection circuit will be enabled; otherwise, it will be turned off.
- It is the user's responsibility to supply the VCCIO voltage that is reported in the PAD Specification File.

BLOCK

Blocks timing analysis on nets, paths, buses, or component pins that are irrelevant to the timing of the design. If a net is specified, the net and all paths through the specified net will blocked. If a bus is specified, the bus, all nets in the bus, and all paths through the nets defined by the bus will be blocked. If a component pin is specified, any path going through the specified pin of the named component will be blocked.

The TRACE report will include all paths that are covered with the BLOCK PATH preference, but it will not include details for a BLOCK NET preference. In TRACE, the signal defined in a BLOCK NET preference is removed from the timing-graph analysis completely. Because there is no path or coverage

related to this signal any longer, all paths through the signal are blocked/ removed and considered covered.

Note

If two input clocks are related according to the board designs, then use CLKSKEWDIFF (which can only be used for port) preference to tell the TRACE that the two clocks are related. Then, when you use the BLOCK FROM CLKNET (/TO CLKNET) preference, Cross Domain Crossing (CDC) paths that under these two input clocks will be blocked.

Otherwise, if these two inputs are unrelated, there is no need to (and user cannot) use BLOCK FROM CLKNET (/TO CLKNET) for these CDC paths.

Device Support All

Syntax BLOCK NET <*net_name*> ;

BLOCK BUS <*bus_name*> ;

BLOCK COMP <*comp_name*> PIN <*pin_name*> ;

BLOCK PATH <*path_spec*> ;

BLOCK <*path_class*> ;

where:

<*path_spec*> ::= [FROM <*obj_type*> <*obj_name*>][TO <*obj_type*> <*obj_name*>]

<*obj_type*> ::= CELL | PORT | CLKNET | ASIC PIN | GROUP

<*obj_name*> ::= identifier

For example, CELL <*identifier*>, PORT <*identifier*>, CLKNET <*identifier*> (ASIC <*identifier*> PIN <*identifier*>), GROUP <*indentifier*>

Note

The object type in BLOCK PATH FROM <obj_type>TO <obj_type> must be of the same category, as follows:

Category 1: PORT/CELL/ASIC PIN/GROUP
Category 2: CLKNET

BLOCK preferences using the following syntax examples are valid, since the FROM and TO object types are from Category 1.

BLOCK PATH FROM CELL TO PORT;
BLOCK PATH FROM PORT TO PORT;
BLOCK PATH FROM GROUP TO PORT;
BLOCK PATH FROM ASIC PIN TO CELL;

But BLOCK preferences using the following syntax are invalid, since the Category 2 CLKNET is mixed with Category 1 object types:

BLOCK PATH FROM CLKNET TO CELL/PORT/GROUP/ASIC PIN;
BLOCK PATH FROM CELL/PORT/GROUP/ASIC PIN to CLKNET;

For Category 2, the path must be from CLKNET to CLKNET.

where:

<*path_class*> can be any one of the following general path classes:

RESETPATHS – Specifies all asynchronous set/reset paths, through an asynchronous set/reset pin on a design component. This is in the "ON" condition by default, so when a new project is created in Diamond, the BLOCK RESETPATHS preference will appear in the project's preference file.

ASYNCPATHS – Blocks all paths from pads to FF inputs for the frequency and period preferences. This is in the "ON" condition by default, so when a new project is created in Diamond, the BLOCK ASYNCPATHS preference will appear in the project's preference file.

JTAGPATHS – Identifies any output signal from "JTCK" of cellmode "JTAG" and blocks all registers driven by this signal when doing FMAX analysis.

RD_DURING_WR_PATHS – Globally blocks RAM read paths during the write operation. This is in the "OFF" condition by default.

The following keyword can also be used with the BLOCK preference but is not recommended, as explained in the Note that follows:

INTERCLOCKDOMAIN PATHS – Specifies all paths involving data transfer between registers that are clocked by different clock nets—even when those clock nets are related.

Note:

Even *without* the BLOCK INTERCLOCKDOMAIN PATHS preference, TRACE does not time data transfers between registers that are clocked by unrelated clock nets. Related clock nets are those that come from different outputs of the same PLL, through different inferred DCS elements, through a CLKDIV, and others. These paths are not generally false paths. When the BLOCK INTERCLOCKDOMAIN PATHS preference is used, these paths are blocked, which is very risky.

JITTER – Blocks jitter from timing analysis. This includes any input clock jitter specified by a PERIOD or FREQUENCY preference, as well as system jitter.

Using Wildcards For the logical BLOCK preference, NET names with wildcard expressions are allowed. PORT or CELL names with wildcard expressions are also allowed. See "Using Wildcard Expressions in Preferences" on page 533 for details about wildcard usage and range expansion.

Examples The following preference command blocks timing analysis on a net named n1:

```
BLOCK NET n1;
```

The following preference command blocks timing analysis on pin A1 of component SLICE_4.

```
BLOCK COMP "SLICE_4" PIN "A1" ;
```

The following preference command blocks timing analysis on all paths terminating at an asynchronous set/reset input of a design component:

```
BLOCK RESETPATHS;
```

The following preference command blocks the path between two ports:

```
BLOCK PATH FROM PORT "LOCAL_CMD0" TO PORT "RAM_RD";
```

The following preference blocks specific interclock domains between clock nets. All maximum frequency (fMAX) paths from clknet_1 to clknet_2 will be blocked from timing analysis.

```
BLOCK PATH FROM CLKNET "clknet_1 " TO CLKNET "clknet_2 ";
```

The following preference blocks all jitter from timing analysis, including all system jitter as well as jitter that has been specified through a PERIOD or FREQUENCY preference:

```
BLOCK JITTER;
```

See Also ▶"Setting TRACE Options" on page 696

CLKDELAY

Sets delays for clock boosting on a CELL (flip-flop).

Device Support LatticeSC/M

Syntax CLKDELAY CELL <*cell_name*><*delay*>;

CLKDELAY GROUP <*group_name*><*delay*>;

where:

<*cell_name*> ::= string

<*group_name*> ::= string

<*delay*> ::= DEL0 | DEL1 | DEL2 | DEL3

The <*cell_name*> and <*group_name*> identifiers must be the flattened logical instance names. Applicable components include instances of flip-flops in PFUs and I/O registers.

The value of DEL0 is no delay. For values of DEL1, DEL2, AND DEL3, refer to the LatticeSC data sheet.

Example The following preference constrains an I/O register for clock boosting:

```
CLKDELAY CELL "my_delay" DEL0
```

CLKSKEWDIFF

Enables TRACE (trce) to take into account the input skew between two top-level, input clock ports. When applied, this preference influences the fMAX-path clock skew crossing over these two clock domains. You specify the targeted input skew delay in a relative manner or as a range. See examples below.

Note

CLKSKEWDIFF can only be applied to input clock skew between two input clock ports and does not support references between different clock nets (CLKNETs). In addition, CLKSKEWDIFF is not applicable to a phase-shifted clock produced by a sysCLOCK PLL. The skew on these clocks are calculated automatically by TRACE and processed in the flow, so there is no need to use a CLKSKEWDIFF preference in these cases.

Device Support All

Syntax CLKSKEWDIFF CLKPORT <*port_name*> CLKPORT <*port_name*> <*max_time_unit*> [MIN <*time_unit*>];

where:

CLKPORT <*port_name*> ::= the name of a top level clock port (IO).

<*max_time_unit*> ::= the maximum skew between two top-level ports. (Refer to Units section “Preference Syntax Guidelines and Conventions” on page 527.)

<*time_unit*> ::= the skew between two top-level ports. (Refer to Units section “Preference Syntax Guidelines and Conventions” on page 527.)

MIN <*time_unit*> ::= specifies the minimum clock skew time between two top-level clock ports.

These keywords for minimum and maximum setup and hold time account for how the skew varies, within some range, between the primary input clocks in actual operating conditions. TRACE reports these min/max values for setup and hold times. This syntax guarantees that correct setup is handled by PAR and that any hold violation is reported. PAR follows TRACE in checking setup times, but it cannot fix hold violations.

Examples In the following example, the clock arrives at "clock1" with a 2 ns delay compared to when it arrives at "clock2." TRACE takes this 2 ns value into account while performing clock skew computation.

```
CLKSKEWDIFF CLKPORT "clock1" CLKPORT "clock2" 2 NS;
```

The following example uses a min/max specification, which allows TRACE to derive a correct number for setup/hold time.

```
clkskewdiff clkport "clka" clkport "clkb" 1.4ns Min 0.4ns;
```

The above example defines that "clka" is lagging behind "clkb" within a range of 0.4ns to 1.4ns. Therefore, the data path from “clka” to “clkb” will use 1.4ns (max) for the setup and 0.4ns (min) for the hold.

The above preference example also defines that “clkb” is lagging before “clka” within a range of 0.4ns to 1.4ns, which is equivalent to the following:

```
clkskewdiff clkport "clkb" clkport "clka" -0.4ns Min -1.4ns;
```

Therefore, the data path from “clkb” to “clka” will use -0.4ns for the setup and -1.4 ns for the hold.

CLKSKEWDISABLE

CLKSKEWDISABLE affects the timing analysis resulting from other preferences such as MULTICYCLE. It disables clock skew computation between two nets. These two nets can be related (originate from the same clock source) or unrelated. Data paths that have the same source and destination clock nets cannot be disabled by using this preference. When CLKSKEWDISABLE is applied to two clock nets, TRACE will always consider the two clock domains related, and it will time the signals between them accordingly. Since the preference also causes the timing analysis to ignore clock skew, any timing analysis done will include only datapath delays.

Device Support All

Syntax CLKSKEWDISABLE CLKNET <*clknetname*> [CLKNET <*clknetname*>];

where:

CLKNET <*clknetname*> ::= the name of source clock net of paths to be disabled from skew calculation in TRACE

Examples The following preference prevents skew calculation by TRACE on any path whose source is clocked by "clock_int1" and whose destination is clocked by "clock_int2." Nets "clock_int1" and "clock_int2" are related clock nets, that is, nets derived from the same source component.

```
CLKSKEWDISABLE CLKNET "clock_int1" CLKNET "clock_int2";
```

The following preference prevents skew calculation by TRACE on any path whose source is clocked by "clock_int1" or whose destination is clocked by "clock_int1".

```
CLKSKEWDISABLE CLKNET "clock_int1";
```

The CLKSKEWDISABLE preference is directional. This means that the following example defines the skew ignored for the datapath from “clkA” to “clkB”:

```
CLKSKEWDISABLE CLKNET “clkA” CLKNET “clkB”;
```

whereas the following example defines the skew ignored for the datapath from “clkB” to “clkA”

```
CLKSKEWDISABLE CLKNET “clkB” CLKNET “clkA”;
```

CLOCK_TO_OUT

Specifies the required timing constraint to meet on FPGA output relative to its clock.

Device Support All

Syntax CLOCK_TO_OUT ([PORT] <*port_name*> | GROUP <*group_name*> | ALLPORTS)
([[MAX] <*time_spec*>] | [OUTPUT_DELAY <*time_spec*>]
[MIN <*time_spec*>]
(CLKNET [=] <*clkin_netname*>) | CLK PORT <*clkout_portname*>)
[FROM <startpoint>]
(CLKPORT [=]<*clkin_portname*>) | [CLKOUT PORT <*clkout_portname*>]
[PLL_PHASE_BACK]
[CLK_OFFSET <*n*> X] ;

where:

PORT <*port_name*>: uses the name of a top level output port to specify the data path.

GROUP <*group_name*>*:* is a given group definition you defined with the DEFINE GROUP preference.

ALLPORTS: designates that all input ports in the given clock port/net path are affected by the time value specified.

MAX (or blank): defines a CLOCK_TO_OUT goal such that the internal timing of the FPGA will produce the output signal by this goal or earlier in time.

MIN: specifies the fastest clock-to-out time for which the board level hold time requirement is still met. MAX is an optional keyword that assists readability by differentiating the MAX clock-to-out requirement from the MIN value.

<time_spec> ::= <number> <time_unit>

(Refer to Units section of Preference Syntax Guidelines and Conventions.)

OUTPUT_DELAY: is an alternate form for defining the MAX constraint goal. Its value is the amount of time that the board and downstream device take from the clock period. If P is the clock period, then the FPGA internal timing must produce the output signal within P – OUTPUT_DELAY time. Another way to look at it is the OUTPUT_DELAY value is the time from the point the data arrives at the output until the next clock edge. Using this keyword (instead of MAX) allows you to change the FREQUENCY preference without having to change the CLOCK_TO_OUT MAX time_spec value requirement.

See the output delay example.

CLKNET=<*clkin_netname*>: uses a net name to specify the reference clock.

FROM <startpoint>: specifies a maximum delay from a register starting point.

CLKPORT=<*clkin_portname*> uses a port name to specify the reference clock.

Note

In some earlier releases, both the CLOCK_TO_OUT preference that specifies either a CLKNET or a CLKPORT in the syntax would yield the same information about the clock path in the Physical Path Details section. That is, TRACE (**trce**) would trace the clock path to the port in both cases. Specifying the CLKNET was for purposes of convenience only. Now, TRACE will stop at the CLKNET specified and it will not go back to report the port unless CLKPORT is specifically used in the preference. Since the behavior of keywords CLKPORT and CLKNET is now differentiated, the enhancement provides more control. See the example at the end of this topic.

CLKOUT PORT <*clkout_portname*>: is the name of the output clock port with respect to the output data delay that needs to be to be measured. This option can be used with both CLKPORT and CLKNET. This is used for source synchronous interfaces. MAX and MIN values are defined relative to the data output and the clock output. As an example, if the MAX value is set to 2ns, this means that the data will be output from the FPGA at most 2ns after the clock outputs (via the CLKOUT PORT) from the FPGA.

PLL_PHASE_BACK reverses the offset direction of the phase delay programmed into a PLL. For example, you might program a PLL to delay the clock phase by 7/8ths of its cycle. Then Trace timing analysis will add 7/8ths of a cycle to the clock insertion delay. However, the PLL_PHASE_BACK keyword directs Trace to subtract 1/8th (7/8 -1 = -1/8) of a cycle from the clock insertion delay.

When two PLL’s are cascaded, the scenarios will become more complex, especially with PLL Bypass. However, as a general rule, if a PLL_PHASE_BACK is applied to a cascaded PLL chain of two and none of the PLL’s are bypassed or with zero phase shift, then the result will be the same as if PLL_PHASE_BACK had been applied to each of them individually.

If one of them is bypassed or with zero phase shift, the effective result will be the same as if PLL_PHASE_BACK had not been applied to that PLL.

PLL_PHASE_BACK gives you the ability to control the desired effect of the phase delay to match your intent. There is a startup one-cycle latency incurred by using this keyword, so design the system accordingly.

CLK_OFFSET <n> X: adjusts the analysis by a multiple of the clock period. The clock_to_out clock path timing will be adjusted by:

<*n*>::= is a floating point number and can be negative.

Since <n> can be a floating point number, this can be used to pick the opposite clock edge for the analysis; the analysis will then adjust automatically to changes to the clock's frequency or period specification. When this option is not specified, the default value of n=0 is assumed, meaning that no adjustment is made and the edge at the pin is the edge used in the analysis.

Note

CLOCK_TO_OUT syntax must specify either a CLKPORT or CLKNET or a syntax error will result. The use of wildcards is supported for port name identifiers.

Using Wildcards For the logical CLOCK_TO_OUT preference, port and register names with wildcard expressions are allowed. See "Using Wildcard Expressions in Preferences" on page 533 for details about wildcard usage and range expansion.

General Syntax Examples The following are examples of general CLOCK_TO_OUT syntax:

```
CLOCK_TO_OUT ALLPORTS 10ns CLKPORT "clkin";

CLOCK_TO_OUT RESP_REQ 80 NS CLKNET = "N28";
```

Examples Using HOLD Value The following examples specify HOLD Values. The use of the keyword MAX before the output delay time unit value is not necessary, since it is implicit. So the following two preferences are equivalent for specifying HOLD values:

```
CLOCK_TO_OUT RESP_REQ MAX 80 ns MIN 60 ns CLKPORT "clkin";

CLOCK_TO_OUT RESP_REQ 80 ns MIN 60 ns CLKPORT "clkin";
```

When TRACE (trce) is run without the -hld option in the following example, where "dataout" is an output data port clocked by a signal from port "clockin", the preference states that the maximum allowable output delay on "dataout" with respect to the output port "clockout" is 4 ns.

```
CLOCK_TO_OUT "dataout" 4NS CLKPORT = "clockin" CLKOUT PORT
"clockout";
```

In the following example, "dataout" is an output data port clocked by a signal from port "clockin." The preference states that the minimum allowable output delay on "dataout" with respect to output port "clockout" is 4 ns. This MIN constraint will be analyzed by Trace when it does Hold time analysis. the MAX constraint will be analyzed by Trace when it does Setup Time analysis.

```
CLOCK_TO_OUT dataout MAX 10NS MIN 4NS CLKPORT = "clockin"
CLKOUT PORT "clockout";
```

You will receive a TRACE (**trce**) syntax error message if you do not also include the MAX delay number in the CLOCK_TO_OUT preference syntax. For example, in the following preference, only the MIN number is specified, resulting in an error:

```
ERROR - trce: line 5: syntax error on, "MIN", in this
preference,
"CLOCK_TO_OUT Q MIN 2 ns CLKPORT CKIN1 CLKOUT PORT CKOUT;"
```

The following preference will set the output delay for all output ports with relation to CLKIN1:

```
CLOCK_TO_OUT ALLPORTS OUTPUT_DELAY 10ns CLKPORT CKIN1;
```

Examples Using CLKPORT and CLKNET Keywords Here are two preferences using both the CLKPORT and CLKNET keywords, showing the corresponding scope of TRACE reporting. In this case, the CLKNET will stop tracing the path before the PLL, so you will not get PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKNET "pll_rxclk" ;
```

The preference will yield the following clock path:

```
Physical Path Details:
       Clock path pll_inst/pll_utp_0_0 to PFU_33:
    Name     Fanout   Delay (ns)          Site               Resource
  ROUTE         49     2.892    ULPPLL.MCLK to     R3C14.CLK0 pll_rxclk
                    --------
                      2.892   (0.0% logic, 100.0% route), 0 logic levels.
```

In this case, the trace is complete back to the clock port resource and provides PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKPORT "RxClk" ;
```

The preference will yield the following clock path:

```
  Clock path RxClk to PFU_33:
    Name     Fanout   Delay (ns)          Site               Resource
  IN_DEL       ---     1.431         D5.PAD to        D5.INCK RxClk
  ROUTE          1     0.843        D5.INCK to   ULPPLL.CLKIN RxClk_c
  MCLK_DEL     ---     3.605   ULPPLL.CLKIN to    ULPPLL.MCLK pll_inst/pll_utp_0_0
  ROUTE         49     2.892    ULPPLL.MCLK to     R3C14.CLK0 pll_rxclk
                     --------
                       8.771   (57.4% logic, 42.6% route), 2 logic levels.
```

Example Using FROM Register/Path Option The following example defines I/O timing relative to source registers using the FROM <register_name> syntax.

```
CLOCK_TO_OUT PORT “dataio*” 11 ns CLKPORT “clkr” FROM r1data_0io_0;
```

The preference will yield the following clock path:

```
  Clock path clkr to data_in_0_IOLOGIC_S:
 Name      Fanout  Delay (ns)      Site             Resource
IN_DEL      ---     0.744        K30.PAD to      K30.PADDI clkr
ROUTE        33     1.968      K30.PADDI to *LOGICB17A.CLK clkr_c
                   --------
                     2.712   (27.4% logic, 72.6% route), 1 logic levels.
```

Examples Using CLK_OFFSET Option The following example uses a CLK_OFFSET of 0.5X:

```
CLOCK_TO_OUT RESP_REQ 80 NS CLKNET = "N28" CLK_OFFSET 0.5X ;
```

An additional half period of the clock will be used in the analysis. This would be appropriate for a negative edge triggered flop driving RESP_REQ. The constraint is that RESP_REQ must update its value at the device pin no later than 80 ns after the rising edge of the clock N28 at its device pin. The CLK_OFFSET of 0.5 X will analyze the negative clock edge after the rising clock edge arrives at the flop from the device pin, meaning that an additional half period will be added to the clock to out path. This example increases the constraint by using CLK_OFFSET.

```
CLOCK_TO_OUT RESP_REQ 80 NS CLKNET = "N28" CLK_OFFSET -0.5X ;
```

This is similar to the previous example, except that the CLK_OFFSET of -0.5 X will analyze the negative clock edge *before* the rising clock edge arrives at the flop from the device pin, which means that half a clock period will be subtracted from the clock to out path. This example relaxes the constraint by using CLK_OFFSET.

See Also ▶"Setting TRACE Options" on page 696

CUSTOM_IDCODE

Specifies a 32-bit value to be used as the JTAG ID of the device when the MY_ASSP feature is enabled.

Device Support MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2

Syntax CUSTOM_IDCODE {HEX | bin} <value);

Examples The following examples show the two available formats.

```
CUSTOM_IDCODE HEX "0000FFFF";
CUSTOM_IDCODE BIN "00001111000011110000111100001111";
```

CUSTOM_IDCODE is a global preference. If it is included more than once in the preference file, only the last instance will be honored.

See Also ▶"SYSCONFIG" on page 1250

DEFINE BUS

Specifies a grouping of nets to be used in the following preferences: BLOCK, MAXDELAY, MAXSKEW, PRIORITIZE.

Device Support All

Syntax DEFINE BUS <*bus_name*> (NET <*net_name*>)+;

Examples The following command defines a bus named busA composed of eight nets named net pen (0-7):

```
DEFINE BUS "busA" NET "pen0" NET "pen1" NET "pen2" NET "pen3"
NET "pen4" NET "pen5" NET "pen6" NET "pen7";
```

DEFINE GROUP

Defines a single identifier that will refer to a group of objects.

PORT/CELL/ASIC groups can be defined using Spreadsheet View or manually in a text editor. For more information, see "Defining Port, Cell, and ASIC Groups" on page 506.

Device Support All

Syntax DEFINE <*obj_type*> GROUP <*group_name*> (ASIC <*identifier*> PIN <*identifier*>)+ | (<*obj_name*>)+ ;

where:

<*group_name*> ::= string

<*obj_type*> ::= CELL | PORT | ASIC

Note

The DEFINE GROUP preference should be used instead of the legacy PIO-based PGROUP preference. Although you can still create a PIO-based PGROUP in the logical preference file using a text editor, this is not recommended. The Diamond preference views do not support the display or editing of PIO-based PGROUPs.

Using Wildcards For the logical DEFINE GROUP preference, wildcard expressions are allowed on CELLs, PORTs, and ASIC PINs. See "Using Wildcard Expressions in Preferences" on page 533 for details about wildcard usage and range expansion.

Note

The use of wildcards is supported for names in the identifier list. All objects named in the group must be of the same type as specified by <*obj_type*>.

General Syntax Examples In the following examples, groups are defined and then used with MULTICYCLE and MAXDELAY preferences.

```
DEFINE CELL GROUP "slow_group" "data*" "other_reg_{0-1}""flop";

DEFINE PORT GROUP "fast_ins" data* adr*;

DEFINE ASIC GROUP "asic1group" ASIC "serdes_fifo0_3" PIN "Q34"
ASIC "serdes_fifo0_3" PIN "Q33";

DEFINE ASIC GROUP "asic2group" ASIC "serdes_fifo3_0" PIN "D1*"
ASIC "serdes_fifo3_0" PIN "D2*";

MULTICYCLE FROM GROUP slow_group 3 X;

MULTICYCLE FROM GROUP fast_ins 1 X;

MAXDELAY FROM GROUP "asic1group" 10 ns;

MULTICYCLE TO GROUP "asic2group" 10 ns;
```

Example Using IOBUF attributes on PORT GROUPs In the following example, IOBUF properties are applied on PORT GROUPS:

```
DEFINE PORT GROUP "grp1" "a1" "a2" "a3";
IOBUF GROUP "grp1"  IO_TYPE=LVCMOS33 PULLMODE=UP BANK=4;
```

During the map process, all of the IOBUF attributes will be attached to individual ports in the physical design (.ncd) file. The DEFINE PORT GROUP will be converted to a PGROUP by the mapper, and the bank assignment on the group "grp1" will have a LOCATE PGROUP in the schematic section. A DEFINE GROUP preference will be included in the section following the schematic section.

The .prf file will have the following preferXences:

```
SCHEMATIC START;
PGROUP "grp1" COMP "a1" COMP "a2" COMP "a3";
            LOCATE PGROUP "grp1" BANK 4;
SCHEMATIC END;
DEFINE PORT GROUP "grp1" "a1" "a2" "a3";
```

Troubleshooting Group Conflicts Redundant DEFINE GROUP declarations in the .lpf file are not handled consistently across “Lattice Diamond” programs and should be avoided. Such multiple definitions of the same group name might appear in the .lpf file because of a semantic error in the original definition. For example, a group created in the .lpf ASCII file might contain a semantic error and not show up in Diamond.

When the software encounters a DEFINE GROUP conflict in the .lpf file, it processes the groups using the following rules:

- Only one group definition is recognized for each group name. When two group definitions with the same name appear in the .lpf file, the software recognizes only the group listed last, whereas the TRACE and PAR programs recognize the group listed first.
- Groups cannot share the same members. When two groups with different names include some of the same elements, the software assigns only non-conflicting elements to the second group.

See Also “Defining Port, Cell, and ASIC Groups” on page 506

“Defining I/O Defaults” on page 475

FBEXTDELAY PCM

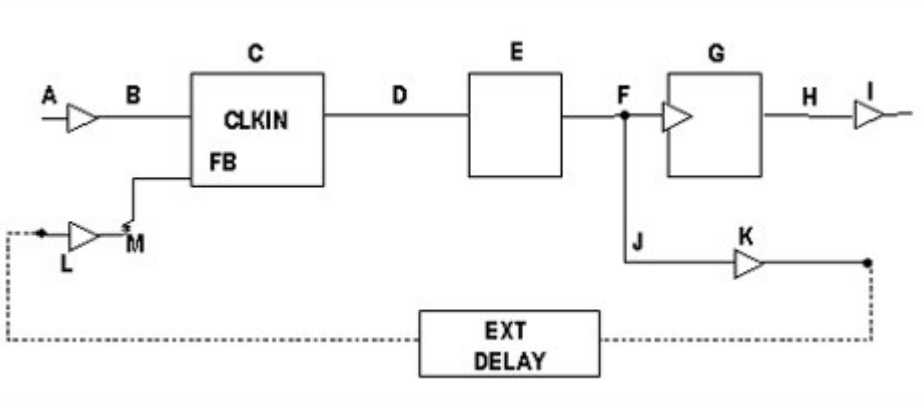
Specifies the external feedback additional delay on a PCM in PLL mode.

Device Support All

Syntax FBEXTDELAY PCM <pcm name> (FROM PORT <*name*> TO PORT <*name*>) <*time*>;

Example `FBEXTDELAY PCM "CLKGEN/PLLDInst_0" FROM PORT "clkout" TO PORT "clkfb" 1 ns;`

Example – Feedback Delay TRACE uses the feedback delay to compensate for the INPUT_SETUP preference.



Feedback delay= C+D+E+J+K+EXT DELAY+L+M

FREQUENCY

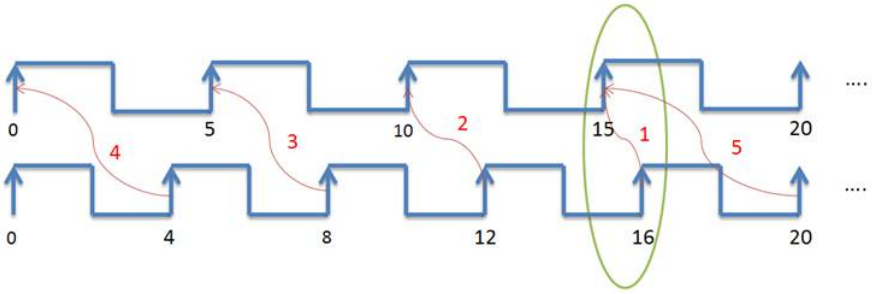
Identifies the minimum operating frequency for all sequential output to sequential input pins clocked by the specified net or port. If no net or port name is given, the preference applies to all clock nets in the design that do not have a specific clock frequency preference.

When two registers are driven by two related clock nets with different frequency requirements, TRACE will automatically use the closest clock-edge gap as the constraint for paths between these two registers. See “Clock Domains Analysis” on page 712 for definition of related clocks.

A delay constraint that is based on different source and destination clocks, with different FREQUENCY preferences defined, is calculated between the closest two active clock edges and is reported as weighted slack in the TRACE report. This can produce a timing error because of the tight timing requirement from the small gap between two clock edges, as shown in the following diagram.

This example shows the clocks aligned at the beginning and an attempt to find the delay constraint in ten cycles of the destination clock. In this case, there are many values: 4ns, 3ns, 2ns, 1ns, 5ns. Trace will pick the worst one, which is 1ns, and use it as delay constraint.

The weighted slack is calculated as destination clock / delay constraint * original slack. If the original slack of this crossing clock domain path (5 ns -> 4



ns) is 0.25 ns, then the weighted slack will be 4/1 * 0.25 = 1 ns. If this is not intentional, you can do one of the following: change the FREQUENCY preferences at the source and destination clocks to adjust the clock-edge gap, use the "MULTICYCLE" on page 1235 preference to define the required time correctly, or use the "BLOCK" on page 1201 preference.

Device Support All

Syntax FREQUENCY <*frequency_value*> [HOLD_MARGIN <*time_spec*>] [CLOCK_JITTER <*time_spec*>];

FREQUENCY NET <*net_name*> <*frequency_value*> MHz | KHz [PAR_ADJ <*par_adj_number*>] [HOLD_MARGIN <*time_spec*>];

FREQUENCY PORT <*clk_port*> <*number*> <*freq_units*> [PAR_ADJ <*par_adj_number*>] [HOLD_MARGIN <*time_spec*>] [CLOCK_JITTER <*time_spec*>];

where:

<*clk_port*> ::= Name of top level clock signal port.

<*freq_units*> ::= MHz | KHz (defaults to MHz)

<*time_spec*> ::= <*number*> <*time_units*>; (defaults to MHz)

<*time_units*> ::= ms, us, ns, ps, fs (defaults to ns)

<*par_adj_number*> ::= Float value for the PAR preference adjustment. This is an absolute value used to adjust the frequency. Defaults to 0.0.

The PAR_ADJ keyword allows you to tighten requirements for placement and routing while preserving the requirements reported by the TRACE static timing analysis tool. The variance in required timing values used in PAR and TRACE allows you to experiment more efficiently with over-constraining your design to determine your best constraint settings for achieving the desired results.

Note

In Diamond, the timing score reported in the PAR report is also based on the actual constraint and not on the PAR_ADJ value.

Constraints will always be tightened when PAR_ADJ is used, and only positive values can be used for this keyword. In addition, you can only use absolute values, not percentages. See the examples below for acceptable usage.

The HOLD_MARGIN keyword allows you to specify a hold margin value per clock domain, in order to avoid possible hold time violations based on design constraints. A hold margin is a time allowance added to the hold time for a path that might be close to failing, as evidenced by timing analysis. Essentially, it allows you to over-constrain the design. When this keyword is present, TRACE will read the user-supplied hold time preferences and apply them to the timing analysis. TRACE will report any path as in violation that fails to meet both the path constraint and the required hold time margin. The hold margin time defaults to 0.0.

For example, TRACE reports a given path with a 1 ps hold margin. It is very likely that a given hold time path that passes at 1 ps will fail on the actual silicon, so a margin is necessary. Prior versions of TRACE would automatically add in a guard band for hold times; however, with the current built-in min/max timing, you should set this margin manually. Generally, adding hold time margins is done with a fixed value such as 200 ps per clock domain. Some clock domains might require less or more of a margin. Hold time margins cannot be achieved by adjusting the fMAX.

The CLOCK_JITTER keyword allows you to specify a peak-to-peak jitter value for the incoming clock and the system jitter affecting the clock port. CLOCK_JITTER is only available for a clock port.

Using Wildcards For the logical FREQUENCY preference, wildcard expressions are allowed for NETs. See “Using Wildcard Expressions in Preferences” on page 533 for details about wildcard usage and range expansion.

Examples The following preference assigns a frequency of 100 MHz to a net named clk1:

```
FREQUENCY NET "clk1" 100 MHz;
```

The following preference assigns a global frequency preference:

```
FREQUENCY 20 MHz;
```

The following preference assigns an adjusted PAR value so that PAR and TRACE timing requirements vary and create the desired results for place-and-route and timing reporting. The keyword PAR_ADJ uses the simple addition formula of 250 + 25 = 275 MHz to direct PAR, while running TRACE on the 250 MHz.

```
FREQUENCY NET "CLK1_c" 250.0 MHz PAR_ADJ 25 ;
```

The following logical preference specifies a hold margin value for each clock domain. It directs that any data path for this preference must meet hold time requirement by at least 1 ns. Those paths that meet the hold time by less than 1 ns will fail in the TRACE report.

```
FREQUENCY NET "RX_CLKA_CMOS_c" 100 MHz HOLD_MARGIN 1 ns;
```

The following logical preference specifies a frequency of 100 MHz to a port named CLK and a clock jitter of 1 ns. The jitter will affect both the rising edge and falling edge timing analysis.

```
FREQUENCY PORT "CLK" 100 MHz CLOCK_JITTER 1 ns ;
```

See Also ▶"SYSTEM_JITTER" on page 1260

▶ "Setting TRACE Options" on page 696

GSR_NET

Assigns the specified net as the input to the global set/reset (GSR) buffer inferred by the design mapper (MAP). The GSR_NET preference allows you to specify the net to be applied as the GSR buffer input if more than one net qualifies to serve as the GSR signal.

GSR is a signal that will initialize all registered elements of the design that have the GSR input enabled. If no GSR buffer circuit is included in the input native generic database (NGD), by default, MAP will infer one based on the set/reset signal that has the most loads. To be considered a legal candidate, the net assigned using the GSR_NET preference must drive at least one registered element's set/reset input, and the element must be enabled to be sensitive to GSR.

MAP will ignore GSR_NET preferences if GSR inferencing is disabled (-noinferGSR).

Device Support All

Usage If a GSR buffer has not been included in the design's HDL or schematic source through the GSR library element, it can be inferred automatically by either logic synthesis or Diamond design mapper (MAP). In most scenarios, it is preferable to either instance the GSR buffer or depend on MAP to introduce a correct GSR buffer circuit.

Syntax GSR_NET NET <*net_name*>;

where:

<*net_name*> is a logical net.

Example If your design has multiple set/reset signals, and the automatic detection by logic synthesis or MAP chooses a GSR driver that's not appropriate for your design, you would use the syntax shown in the following example to explicitly assign a net to drive the GSR buffer circuit:

```
GSR_NET NET "rst_pcie";
```

HGROUP

Hierarchical grouping constraint. HGROUP is used to infer a unique placement group for each instance of an object from the logical domain, and it is commonly defined as part of a VHDL architecture or a Verilog module declaration. It indicates a logical partition of a group of components, which gets translated into a physical partition for placement and routing (PAR). HGROUPs are translated by the design mapper into PGROUPs, which direct the placer algorithm of PAR to place the group members in proximity.

The universal grouping constraint, UGROUP, is an alternative to HGROUP and is used to infer a single placement group composed of all instances of an object.

Device Support All

Usage You cannot enter an HGROUP directly in the Diamond preference-editing tools. You must specify an HGROUP definition as a logical preference in the logical preference file (.lpf) or as an HDL attribute in the HDL source files. If you place an HGROUP in an .lpf file, you will need to use the module name and hierarchy that is generated by the synthesis netlist. The reason is that the post-synthesis module name and hierarchy might be different than the name in the HDL. All the subsequent instances will have the HGROUP identifier as part of the normal elaboration performed by logic synthesis.

Note

Synthesis directives to preserve hierarchy are typically needed to retain the instances constrained by the HGROUP attribute.

In the design mapping phase, PGROUPs that are inferred from HGROUPs in the post-synthesis netlist appear in the schematic section of the physical preference file (.prf). Those inferred from the .lpf file appear in the section following the schematic section.

Syntax HGROUP <*hgroup_name*> [TYPE <*type_name*>] [BBOX <height width>];

where:

<*hgroup_name*> is a user-defined name of the HGROUP.

TYPE <*type_name*> Identifies the name of the type of block you are grouping. A given block type can only be assigned membership to one group, either an

HGROUP or an UGROUP. It cannot be in two or more groups at the same time.

BBOX *<height_width>* is used to optionally define the maximum height and width of the group's bounding box. Without a BBOX parameter, the place and route program will default to the minimal square that will accommodate the design logic.

Example Assume that you have defined a top-level block known as "ANEB2" that is instanced four times as "b1" "b2", "b3" and "b4." The .lpf file contains the following preferences:

```
HGROUP "h1" TYPE "ANEB2";
UGROUP "u1" BBOX 2 3 BLKNAME "b1" BLKNAME "b2";
```

This should result in an error condition, because "b1" and "b2" are part of the HGROUP "h1".

Now let us assume that the illegal UGROUP "u1" is removed from the example, and your HGROUP is as follows:

Figure 34:

```
HGROUP "h1" TYPE "ANEB2" BBOX 3 4;
```

After the design is mapped, your resulting .prf file will contain the following PGROUP in the schematic section:

```
PGROUP "h1" BBOX 3 4 DEVSIZE
    COMP "binst/SLICE_0"
    COMP "binst/SLICE_1"
    COMP "binst/SLICE_2"
    COMP "binst/SLICE_3";
```

HGROUPs and Diamond Preference Tools When Diamond encounters an HGROUP, it automatically infers one or more UGROUP instances and presents them as UGROUPs in the Group sheet of Spreadsheet View. If you modify a UGROUP, Diamond will add a new UGROUP declaration to the .lpf file and override the original inferred group.

The HGROUP preference can be applied on a particular block TYPE. All the subsequent block instances of the same type will have the HGROUP identifier. The identifier is then made unique for all the instances.

A given block can be assigned as a member of only one group, either an HGROUP or a UGROUP. See "Troubleshooting UGROUP Conflicts" on page 1265 for information about how the software processes conflicting groups that might appear in the .lpf file.

See Also ▶"UGROUP" on page 1263 (preference)

▶ "UGROUP" on page 1329 (HDL attribute)

I2C ADDRESS

Every slave device, connected to the I2C bus, must have an unique I2C address. The I2C address can be either 7-bits or 10-bits long.

In 7-bit I2C Addressing the first byte (immediately after the START condition) contains the I2C slave address. The I2C address is 7-bits long. It is transmitted in the seven most significant bits (MSB). The last (eighth bit) of the I2C address byte is a data direction bit - a 'zero' indicates a transmission (I2C WRITE), a 'one' indicates a request for data (I2C READ).

With 7-bit addressing only 112 I2C slave addresses are available. To prevent address clashes Philips Semiconductors (now NXP Semiconductors) has introduced a 10 bit address scheme. Devices with 7-bit and 10-bit addresses can be connected to the same I2C-bus.

The 10-bit address is transmitted within the first two bytes following a START condition or a repeated START condition.

Five most significant bits of the first address byte are predefined (1111 0). I2C slave devices with 7-bit addressing ignore transactions with the first byte in the form of '1111 0XXX'.

The I2C address occupies bits 5 and 6 of the first byte and the eight bits of the second byte.

The transfer direction (read or write) is specified in the eighth bit of the first byte. As in the case of 7-bit I2C address, a 'zero' indicates a transmission (I2C WRITE), a 'one' indicates a request for data (I2C READ).

Slave devices with 10-bit I2C addressing will react to a general call in the same way as slave devices with 7-bit I2C addressing. I2C master devices can transmit the 10-bit I2C address after a general call.

The primary I2C port of the CrossLink device can be used as a user I2C port function or as a device programming port. When used for device programming, the primary I2C port is a slave I2C with a default slave address of 7'b1000000 or 10'b1111000000. The primary I2C port must be enabled in order to support the device programming using the I2C protocol.

Device Support All

Syntax I2CADDRESS 7/10BIT <*binary_string*>;

where:

<*binary_string*> ::= binary value of up to 10 bits

Example `I2CADDRESS 10BIT "11111000";`

INPUT_SETUP

Specifies a setup time requirement for input ports relative to a clock net.

Device Support All

Syntax INPUT_SETUP (PORT <*port_name*> | GROUP <*group_name*> | ALLPORTS)
(<*time_spec*> | INPUT_DELAY <*time_spec*>)
[HOLD <*time_spec*>]
((CLKPORT [=] <*clk_portname*>) | (CLKNET [=] <*clk_netname*>))
[PLL_PHASE_BACK]
[CLK_OFFSET <n> X]
[SS];

where:

PORT <*port_name*>: uses the name of a top-level port to specify the data path.

GROUP <*group_name*> is a given group definition you defined with the DEFINE GROUP preference.

ALLPORTS designates that all input ports in the given clock port/net path are affected by the time value specified.

<*time_spec*> ::= <*number*> <*time_unit*>

<*time_unit*> ::= MS | US | NS | PS | FS

(Refer to Units section of Preference Syntax Guidelines and Conventions.)

INPUT_DELAY is an alternate form to define this input timing constraint, where the time value is independent of the frequency/period. As an example, the two constraints below are equivalent if the clock period is 10ns, since 4 = 10-6. For hold time, the software will take the negative of the value you have set for HOLD <*time_spec*> as the HOLD offset.

```
INPUT_SETUP in_data1 INPUT_DELAY 6 CLKPORT clk;
INPUT_SETUP in_data1 4 CLKPORT clk;
```

If the clock period constraint value changes, the input constraint that uses the INPUT_DELAY keyword does not need to be changed, whereas the input constraint that does not use INPUT_DELAY should be changed.

See the example for INPUT_DELAY.

HOLD <time_spec>: specifies the hold time.

CLKPORT=<*clkin_portname*> uses a port name to specify the reference clock.

CLKNET=<*clkin_netname*> uses a net name to specify the reference clock.

PLL_PHASE_BACK reverses the offset direction of phase delay programmed into a PLL. For example, you might program a PPLL/HPPLL to delay the clock phase by 7/8ths of its cycle. Then the Trace Timing Wizard™ will add 7/8ths of a cycle to the clock insertion delay. However, The PLL_PHASE_BACK

keyword directs the Trace Timing Wizard to subtract 1/8th (7/8 -1 = -1/8) of a cycle from the clock insertion delay.

When two PLL's are cascaded, the scenarios will become more complex, especially with PLL Bypass. However, as a general rule, if a PLL_PHASE_BACK is applied to a cascaded PLL chain of two and none of the PLL's are bypassed or with zero phase shift, then the result will be the same as if PLL_PHASE_BACK had been applied to each of them individually. If one of them is bypassed or with zero phase shift, the effective result will be the same as if PLL_PHASE_BACK had not been applied to that PLL.

CLK_OFFSET <n> X: adjusts the analysis by a multiple of the clock period. The input_setup path timing will be adjusted by:

<*n*> ::= floating point number and can be negative.

Since <*n*> can be a floating point number, this can be used to pick the opposite clock edge for the analysis. The analysis will adjust automatically to changes to the clock's frequency or period specification. If this option is not specified, the default value of n=0 is assumed, meaning that no adjustment is made, and the edge at the pin is the edge used in the analysis. One use of this is in the analysis of a set of input registers on a clock domain where some registers are positive edge active and others are negative edge active.

SS: indicates that the data input and clock input are source-synchronous. This allows input jitter specified on the clock to be ignored for setup and hold timing analysis.

Using Wildcards For the logical INPUT_SETUP preference, PORT names with wildcard expressions are allowed. See "Using Wildcard Expressions in Preferences" on page 533 for details about wildcard usage and range expansion.

Examples The following examples show usage that specifies absolute time values:

```
INPUT_SETUP ALLPORTS 10 NS CLKNET="n28";
INPUT_SETUP LOCAL_CMD* 10 ns CLKNET="n28";
INPUT_SETUP "a_1" 4 ns CLKNET "clk3";
```

Given a PERIOD and INPUT_SETUP preference that uses the INPUT_DELAY keyword to place a specific relative time value that is frequency/period dependent, the preference might appear as follows:

```
PERIOD PORT "clk" 104 ns ;
INPUT_SETUP PORT "in1" INPUT_DELAY 6 ns CLKPORT "clk" ;
INPUT_SETUP PORT "in2" 4 ns CLKPORT "clk";
```

Both in1 and in2 have the same input constraint.

The following example uses the CLK_OFFSET option.

```
INPUT_SETUP PORT "RESP_REQ" 4 NS CLKPORT = "clock" CLK_OFFSET
0.5X ;
```

An additional half period of the clock will be used in the analysis. This would be appropriate for a negative edge triggered flop that is capturing the input from port RESP_REQ. The constraint is that RESP_REQ will arrive at the device pin at least 4ns prior to rising edge of "clock", and the analysis will test whether the data arrives at the capturing flop in enough time prior to the negative edge of the clock signal at the flop. 0.5 X will analyze the negative clock edge after the rising clock edge arrives at the flop from the device pin, meaning an additional half period will be added to the clock to out path.

This next example also uses the CLK_OFFSET option.

```
INPUT_SETUP PORT RESP_REQ 4 NS CLKPORT = "clock" CLK_OFFSET
0.5X ;
```

In this example, -0.5 X will analyze the negative clock edge before the rising clock edge arrives at the flop from the device pin, which means that half a clock period will be subtracted from the clock to out path.

The following example uses the SS option.

```
INPUT_SETUP PORT "datain_0" 0.480000 ns HOLD 0.480000 ns
CLKPORT "CLK" SS;
```

In this example, any input jitter specified on clock port "CLK" will be ignored for both INPUT_SETUP and HOLD analysis.

See Also ▶"PERIOD" on page 1239

- "FREQUENCY" on page 1216
- "CLOCK_TO_OUT" on page 1208
- "Setting TRACE Options" on page 696

IOBUF

Defines I/O components.

Device Support All

Note

The only preferences that are not written to the physical preference file (.prf) are IOBUF preferences. This information resides in the physical design (.ncd) file only.

Syntax IOBUF [ALLPORTS] (<*keyword*>=<*value*>)+;

IOBUF [PORT <*port_name*> | GROUP <*group_name*>]
[VREF=<*vref_name*>](<*keyword*>=<*value*>)+;

IOBUF GROUP <*group_name*> (*group_keyword*=*value*)+;

[LOCATE VREF <*vref_name*> SITE <*site_name*>;]

where:

<*port_name*> = string. These are not the actual top-level port names but the signal name attached to the port. PIOs in the physical design (.ncd) file are named using this convention. Any multiple listings or wildcards should be done using GROUPs.

<*group_name*> = string

<keyword> (LatticeECP/EC/XP) ::= IO_TYPE, DRIVE, PULLMODE, SLEWRATE.

group_keyword = string.

<*site_name*> = string

<*vref_name*> = string <*keyword*> (LatticeECP3) ::= IO_TYPE, OPENDRAIN, DRIVE, PULLMODE, PCICLAMP, SLEWRATE, TERMINATEVTT, DIFFRESISTOR, DIFFDRIVE, MULTDRIVE, EQ_CAL

For the IOBUF ALLPORTS preference, any possible sub-attributes of IOBUF, such as IO_TYPE, SLEWRATE, and DIFFCURRENT, will only take effect on buffers that have no other specific IOBUF preferences placed upon them. This is because of the preference precedence rule: more specific preferences override more generally applied ones. For example, in an .lpf file there is a set of the following preferences:

```
IOBUF ALLPORTS SLEWRATE="FAST";
IOBUF PORT ichk IO_TYPE=HSTL15_II SLEWRATE="NA";
IOBUF PORT bchk IO_TYPE=HSTL15_II ;
IOBUF PORT ochk IO_TYPE=LVDS SLEWRATE="NA";
```

The global SLEWRATE "FAST" override in this example will not be applied to the I/O buffer port named "bchk," because there is an explicit IOBUF preference ascribed to it. Consequently, the software will interpret any possible missing sub-attributes by their default conditions. So, a SLEWRATE=SLOW is applied in the resulting .ncd file for the "bchk" buffer. To get the desired results of a fast slew rate, you would have to edit that one particular preference to include the explicit slew rate value on that line. For example,

```
IOBUF PORT bchk IO_TYPE=HSTL15_II SLEWRATE="FAST";
```

The precedence rule applies to the preferences as a whole and not on a per-attribute basis. You cannot use a global (ALLPORTS) preference to override part of a specific IOBUF PORT preference. See “Preference Conflict Resolution” on page 395.

Using Wildcards Wildcards can be used with port names to create a DEFINE PORT GROUP preference, after which IOBUF attributes can be assigned to the entire group. The use of the “*” wildcard, which serves a purpose similar to the ALLPORTS preference, can be used with DEFINE PORT GROUP to assign I/O defaults for all ports in the design. See “Defining I/O Defaults” on page 475.

```
DEFINE PORT GROUP "bank1" "in*" "out_[0-31]";
IOBUF GROUP "bank1" IO_TYPE=SSTL18_II IMPEDANCE=33
TERMINATEVCCIO=120 TERMINATEGND=120;

DEFINE PORT GROUP "all_ports" "*" ;
IOBUF GROUP "all_ports" IO_TYPE=LVCMOS18 PULLMODE=DOWN ;
```

See “Using Wildcard Expressions in Preferences” on page 533 for details about wildcard usage and range expansion.

See Also ▶“IOBUF Attributes” on page 478

▶ “DEFINE GROUP” on page 1213

LOCATE

When applied to a specified component, places the component at a specified site or bank and locks the component to the site or bank. This means that the component will not be unplaced, moved, swapped, or deleted.

When applied to a specified group of instances, places the group at a specified site or within a region. Refer to the “UGROUP” on page 1263 preference topic.

Device Support All

Syntax LOCATE COMP <*comp_name*> [SITE<*site_name*> | BANK<bank_num;] LOCATE [UGROUP <*ugroup_name*>] [SITE <*site_name*> | REGION <*region_name*>;]

where:

<*comp_name*> ::= string
<*site_name*> ::= string
<*bank_num*> ::= number
<*ugroup_name*> ::= string
<*region_name*> ::= string

You can LOCATE a given UGROUP within a specified region. The placer can place it anywhere within the boundary of the region. If the region is not defined in the .lpf file before the LOCATE UGROUP definition, you will encounter error messages.

You can also LOCATE a given UGROUP at a particular site. The specified site can be a slice site or an EBR site. If you use a slice site, the slice name must end with the letter D; for example, R2C2D. Slice sites come in groups of four, and the four site names end with the letters A, B, C, and D respectively.

This specified site will become the upper left corner anchor location where the logic in the UGROUP will be placed.

Examples

```
LOCATE COMP "A" SITE "11" ;  // IO site

LOCATE COMP "A" BANK 2 ;

LOCATE COMP "inst1/PLLInst_0" SITE "PLL_R43C5" ;

LOCATE UGROUP "ugroup_0" SITE "R2C2D" ;//note that 'D' site is
the anchor

REGION "my_region" "R3C3D" 17 9;

LOCATE UGROUP "xyz_ugroup" SITE "my_region" ;
```

If the <*comp_name*> or <*site_name*> begins with anything other than an alpha character (for example, "11C7"), you must enclose the name in quotes.

See Also ▶"LOCATE VREF" on page 1228

LOCATE VREF

Assigns a PIO site that will serve as the input pin for an on-chip voltage reference. Referenced input voltage pins are required when implementing an externally referenced signaling standard such as HSTL or SSTL. After the Vref location has been established, it can be associated with a port group.

Device Support All

Syntax LOCATE VREF <*vref_name*> SITE <*site_name*> [RAIL <*rail_number*>] IOTYPE <iotype value>;

where:

<*vref_name*> ::= string

<*rail_num*> ::= integer or [Vref1|Vref2] (LatticeSC/M only)

<*iotype_value*> ::= [HSTL18_I | HTSL18_II | HSTL15_I | SSTL33_I | SSTL3D_II | SSTL25_I, SSTAL25_II, SSTL18_I | SSTL18_II]

For more details regarding I/O type, see the sysIO Usage Guide for your target device family.

For LatticeSC/M devices, each bank can support up to two separate VREF input voltages, VREF1 and VREF2, that set the threshold for the referenced input buffers. Any I/O in the bank can potentially be used to input a VREF voltage. Once the I/O pin within the bank becomes the external VREF input pin, it is no longer available for use as an I/O. If differential outputs that require external bias are implemented in a bank, then the VREF1 of that bank will be used to provide this bias. This VREF1 pin can no longer be available as a VREF pin. For more information, see the LatticeSC PURESPEED I/O Usage Guide, TN1088)

Note

The LOCATE VREF <*vref_name*> preference is not available for MachXO devices.

The LOCATE VREF <*vref_name*> RAIL <*rail_num*> preference is available only for LatticeSC/M devices.

Example – Associate VREF by Signaling Standard This LOCATE preference, in the .lpf file, establishes the N21 pin as a voltage reference named VREF1_BANK_3:

```
LOCATE VREF "VREF1_BANK_3" SITE "N21" IOTYPE "SSTL18_I";
```

The following DEFINE PORT GROUP “d1_group” and IOBUF GROUP create a user-defined alias for d1_0-d1_7 and assign the port group signal standard SSTL18_I.

```
DEFINE PORT GROUP "d1_group" "d1_0"
"d1_1"
"d1_2"
"d1_3"
"d1_4"
"d1_5"
"d1_6"
"d1_7" ;
IOBUF GROUP "d1_group" IO_TYPE=SSTL18_I ;
```

After map the .prf file will contain the following preferences.

Example – Associate VREF with a PORT GROUP This LOCATE preference, in the .lpf file, establishes the D11 pin as a voltage reference named vref_a:

This example, in the .lpf file, defines a port group as “group_a” and assigns it the “vref_a” voltage reference:

```
SCHEMATIC START;
PGROUP "group_a" VREF "VREF1_BANK_3"
COMP "d0_4"
COMP "d0_3"
COMP "d0_2"
COMP "d0_1"
COMP "d0_0";
SCHEMATIC END;
LOCATE VREF "VREF1_BANK_3" SITE "N21";
```

```
LOCATE VREF "vref_a" SITE "D11" ;
```

```
DEFINE PORT GROUP "group_a" "d0_0"
"d0_1"
"d0_2"
"d0_3"
"d0_4" ;
IOBUF GROUP "group_a" IO_TYPE=SSTL18_I VREF=vref_a ;
```

Diamond displays this information in Spreadsheet View. The Spreadsheet View's Port Assignments sheet enables you to set up a Vref location and afterwards assign it to an input port or port group by simply right-clicking in the Vref column.

After map, the .prf file will contain the following preferences:

```
SCHEMATIC START;
PGROUP "group_a" VREF "vref_a"
  COMP "d0_4"
  COMP "d0_3"
  COMP "d0_2"
  COMP "d0_1"
  COMP "d0_0";
SCHEMATIC END ;
COMMERCIAL ;
LOCATE VREF "vref_a" SITE "D11" ;
LOCATE VREF "vref_b" SITE "Y7" ;
DEFINE PORT GROUP "group_a" "d0_0"
"d0_1"
"d0_2"
"d0_3"
"d0_4" ;
```

See Also ▶TN1262, *ECP5 sysIO Usage Guide*

- TN1056, *LatticeECP/EC and LatticeXP sysIO Usage Guide*
- TN1088, *LatticeSC PURESPEED I/O Usage Guide*
- TN1102, *LatticeECP2/M sysIO Usage Guide*
- TN1177, *LatticeECP3 sysIO Usage Guide*

- TN1136, *LatticeXP2 sysIO Usage Guide*
- TN1091, *MachXO sysIO Usage Guide*
- TN1202, *MachXO2 sysIO Usage Guide*
- TN1280, *MachXO3L sysIO Usage Guide*
- TN1305, *CrossLink sysI/O Usage Guide*
- FPGA TN-02065, *Implementing High-Speed Interfaces with MachXO3D Device*

MAXDELAY

Identifies a maximum total delay for a net, bus, or path in the design. When a net is specified, the maximum delay constraint applies to all driver-to-load connections on the net. When a path is specified, the delay value is the constraint for the path including net and component delays as defined by the pathspec rules.

Note

When the path is from register to register, the constraint time is the cycle time of frequency. The software will automatically take care of the half-cycle if the polarity at the source register is different from the destination register.

When a bus is specified, the delay value is for driver-to-load connections on the nets that belong to the specified bus.

Device Support All

Syntax MAXDELAY <*path_spec*> <*time_spec*> [DATAPATH_ONLY] MIN <*value*>ns;

MAXDELAY [NET <*net_name*> | BUS <*bus_name*> | ALLPATHS | ALLNETS] <*time_spec*>;

where:

<*path_spec*> ::= FROM <*path_elem*> [TO <*path_elem*>] | TO <*path_elem*>

Note

It is recommended that you always specify both the start point and end point of the path. If you do not select a component for both FROM and TO, the default for the unspecified path element will include all I/O ports and registers, which is similar to the preference FREQUENCY. If the BLOCK ASYNCPATH preference is set to ON, all input-pad-to-register paths will be removed for the unspecified start point or end point, in the same way as the preference FREQUENCY

<*path_elem*> ::= <*obj_type*> <*obj_name*>
| ASIC <*block_name*> PIN <*pin_name*>

<*obj_type*> ::= CELL | PORT | GROUP

<*obj_name*> ::= identifier

<*block_name*> ::= identifier

<*pin_name*> ::= identifier

<*time_spec*> ::= <*number*> <*time_unit*> (Refer to the units section of Preference Syntax Guidelines and Conventions.)

DATAPATH_ONLY ::= This is an optional parameter to report only data path delay information. When the DATAPATH_ONLY keyword is used, the preference will only include the data path delay and clock skew and/or a setup/hold requirement will be ignored.

MIN <*value*>ns ::= Sets minimum delay value for hold analysis to allow for bus skew.

Note

The MIN keyword is not supported for NET, BUS, ALLNETS, and ALLPATHS.

The ALLNETS | ALLPATHS command establishes a maximum total delay for all nets or all paths in the design. If ALLNETS is specified, the maximum delay constraint applies to all driver-to-load connections on all nets. If ALLPATHS is specified, the delay value is the constraint for all paths.

The MIN keyword uses a time specification parameter, such as 1 ns, as a directive to TRACE timing analysis. It instructs TRACE to use this value for hold analysis when it is doing a hold check for the MAXDELAY preference.

MAXDELAY can be applied only to registers, block pins and other I/Os. It should not be applied to QN outputs of a flip-flop. Since no QN output exists in hardware, the flip-flop is decomposed into using the non-inverted output driving an inverter. If a MAXDELAY is attached to the flip-flop, then both decomposed elements will inherit the MAXDELAY preference.

Note

There might be times when you need to apply a timing preference, such as MULTICYCLE or MAXDELAY, that requires specification of CELL or ASIC; but you are not sure whether the specific instance is an ASIC or a CELL. In these situations, run the Map Report, which lists all ASICS in the “ASIC Components” section. All other registers in the report are CELLs.

Using Wildcards For the logical MAXDELAY preference, CELL and PORT names with wildcard expressions are allowed. See “Using Wildcard Expressions in Preferences” on page 533 for details about wildcard usage and range expansion.

Examples

```
MAXDELAY FROM CELL "BLKWR_SM_1/state_reg4"
 TO CELL "R5_MUX_1/RAM_OE_N_reg" 15 NS;

MAXDELAY FROM PORT "LOCAL_CMD0" TO CELL
 "R5_MUX_1/RAM_OE_N_reg" 16 NS;

MAXDELAY FROM CELL "BLKWR_SM_1/state_reg4"
 TO PORT "RAM_RD" 17 NS;

MAXDELAY FROM PORT "LOCAL_CMD0" TO PORT "RAM_RD" 18 NS;

MAXDELAY FROM ASIC block_8x850 PIN D0
 TO CELL THIS_IN 12 ns;

MAXDELAY FROM GROUP "asic1group" 10 ns;
```

This preference sets a minimum delay value of 2 nanoseconds for hold analysis.

```
MAXDELAY FROM PORT "a1" to PORT "d1" 5 ns MIN 2 ns ;
```

See Also ►"Setting TRACE Options" on page 696

MAXSKEW

Specifies a maximum signal skew between a driver and the driver's loads on a specified clock signal. Skew is the difference between minimum and maximum load delays on a clock net. If no signal is specified, MAXSKEW applies to all signals that have clock pins as loads and do not have a specified skew preference.

Device Support All

Syntax MAXSKEW [NET *<net_name>* | PORT *<port_name>* | BUS *<bus_name>*] *<time_spec>* [CLOCKLOAD_ONLY];

If no net is specified, the MAXSKEW constraint applies to all nets.

where:

<net_name> | *<bus_name>* | *<port_name>* ::= identifier

<time_spec> ::= *<number>* *<time_unit>* (Refer to "Preference Syntax Guidelines and Conventions" on page 527.)

BUS ::= The BUS keyword can be used with MAXSKEW for phase aligning two clocks from the same source, such as a PLL, and meeting a maximum skew between them at the destination registers.

CLOCKLOAD_ONLY ::= This keyword filters out the TRACE report to show only clock destination skews, meaning that it provides an analysis of the clock tree for clock loads only. A clock load means an end point that is used as a clock of a component such as register and does not include a clock generator or mux input.

Examples The following command assigns a maximum skew of 5 nanoseconds to a net named NetB:

```
MAXSKEW NET "NetB" 5 NS;
```

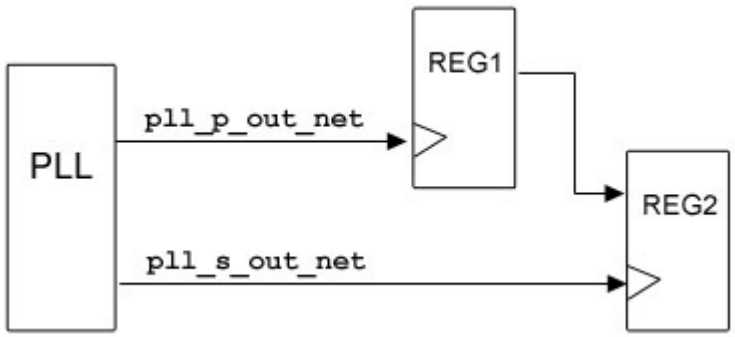
In the following PLL example, "busA" has been defined through the DEFINE BUS preference as

```
DEFINE BUS "busA" NET "pll_p_out_net" NET "pll_s_out_net"
```

This BUS attribute is then used with MAXSKEW to constrain the worst case skew between CLKNET "pll_p_out_net" and "pll_s_out_net." This controls the routing delay for all clock loads. The following command will constrain the skew between Clock nets pll_p_out_net and pll_s_out_net to 1.1 ns:

```
MAXSKEW BUS "busA" 1.1 NS CLOCKLOAD_ONLY;
```

Figure 35:




```
MAXSKEW PORT "NetP" 10 NS;
```

```
MAXSKEW PORT "net1" 4 NS CLOCKLOAD_ONLY;
```

MULTICYCLE

Allows for relaxation of previously defined PERIOD or FREQUENCY constraints on a path.

The MULTICYCLE preference applies only to paths that are covered by the FREQUENCY and PERIOD constraints. It is considered in conjunction with the FREQUENCY and PERIOD constraints specified and will relax ONLY those paths that are constrained by FREQUENCY and PERIOD. Specifying a MULTICYCLE preference without a FREQUENCY or PERIOD preference will not result in paths being timed.

Paths that are not within the domain of the FREQUENCY and PERIOD preference should be constrained by MAXDELAY and not MULTICYCLE.

Device Support All

Recommendation for maximum efficiency: Try to specify all path-based timing constraints in terms of FREQUENCY, MULTICYCLE, CLOCK_TO_OUT, and INPUT_SETUP. If you cannot specify the desired constraints in terms of these preferences, then only use the MAXDELAY and BLOCK PATH preferences where absolutely necessary.

Specifying Multiples Less Than or Equal to Zero for Hold Analysis The MULTICYCLE preference allows you to specify a multiple that is less than or equal to zero. This is useful for hold time analysis.

A typical timing analysis setup time is calculated from a source clock to the next clock cycle of the destination clock. A hold time is calculated from a source clock to the same clock cycle of the destination clock. The MULTICYCLE preference can select how many clock cycles are required for a given path. For example, 2x will allow 2 clock cycles for setup. Therefore, a 0x

will use 0 clock cycles for setup, which simply calculates clock skew. A negative value such as -1x will subtract a clock cycle from the clock skew.

Traditionally, MULTICYCLE is not used for hold analysis. With this feature, the same logic applies to hold analysis, but since hold analysis already only looks at clock skew, a 0x will subtract a clock cycle from the analysis. A -1x will subtract 2 clock cycles from the analysis. This ability to specify less than or equal to zero multicycles should only be used to block paths from timing analysis that are known to have desirable timing. It should only be used for filtering the reports and not for driving place and route.

Syntax MULTICYCLE
[FROM <*path_elem*>] [CLKNET <*Snet*>] [<*path_types*>]
[TO <*path_elem*>][CLKNET <*Dnet*>] <*multiplier*> | <*number NS*>;

where:

<path_elem> :==<obj_type> <obj_name> | ASIC <block_name> PIN <pin_name>

<*obj_type*> ::= CELL | PORT | GROUP

<*obj_name*> ::= identifier

<*block_name*> ::= identifier

<*pin_name*> ::= identifier

<*path_type*> ::= SAMECLKEN | CLKEN_NET <*net_name*> net

SAMECLKEN: the multicycle paths are defined at the source/destination register controlled by the same clock-enable signal.

CLKEN_NET: the multicycle paths are defined at the source/destination register controlled by a given <net_name>.

<*multiplier*> ::= <*factor*> <X | X_SOURCE | X_DEST>

<*multiplier*> is a multiple of the clock period for the clock driving the comp(s). Destination clock period will be applied to the factor when X or X_DEST is used. Source clock period will be applied when X_SOURCE is used.

<*number NS*> is a delay value for the path in nanoseconds.

Note

There might be times when you need to apply a timing preference, such as MULTICYCLE or MAXDELAY, that requires specification of CELL or ASIC; but you are not sure whether the specific instance is an ASIC or a CELL. In these situations, run the Map Report, which lists all ASICS in the “ASIC Components” section. All other registers in the report are CELLs.

Using Wildcards For the MULTICYCLE preference, PORT or CELL names with wildcard expressions are allowed. Like the physical MULTICYCLE preference, this accepts optional CLKNET Snet as Dnet. See “Using Wildcard Expressions in Preferences” on page 533 for details about wildcard usage and range expansion.

Examples When specifying source and destination clock domains, use the keyword CLKNET followed by the clock net name. The multicycle preference will be applied only if the source and destination clock matches the specification. Clock domains can be looked upon as added qualifiers to multicycle.

When nX is used, the constraint is shifted by (n-1) times the source or destination clock period, depending on which keyword is used. When 1X is used, the constraint is shifted by (1 - 1) * period = 0, which makes it the same as in a FREQUENCY preference. When 3X is used, the constraint is shifted by (3 - 1) * period.

In the following command, the X keyword equals X_DEST and uses a factor of 2. If the source clock period is 20 ns and the destination clock period is 40 ns, then the timing constraint for this preference is 20 + (2-1) * 40 = 60 ns.

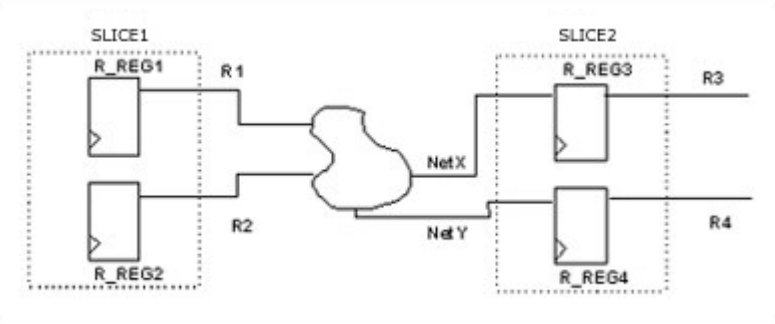
```
MULTICYCLE FROM CLKNET "clk1" TO CLKNET "clk2" 2 X ;
```

The following command will multicycle from R_REG1 to R_REG3 but not from R_REG2.

```
MULTICYCLE FROM CELL "R_REG1" TO CELL "R_REG3" 2 X ;
```

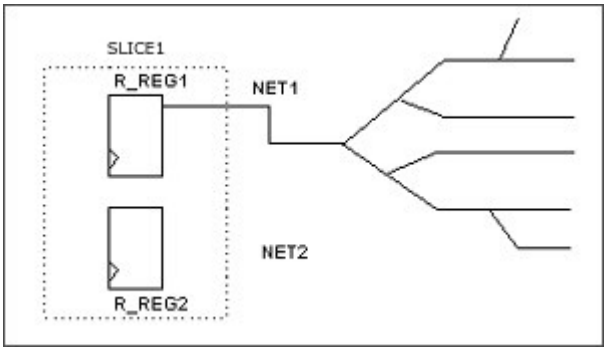
The following command will multicycle from R_REG1 to R_REG3 but not to R_REG4.

```
MULTICYCLE FROM CELL "R_REG1" TO CELL "R_REG3" 2 X ;
```



The following command will multicycle from R_REG1 to all.

```
MULTICYCLE FROM CELL "R_REG1" 2 X ;
```




```
MULTICYCLE FROM GROUP slow_group 3 X;
MULTICYCLE FROM GROUP fast_ins 1 X;
MULTICYCLE TO GROUP "asic2group" 10 ns;
```

```
MULTICYCLE FROM ASIC block_8x850 PIN D0 CLKNET clk100 8 ns;

MULTICYCLE FROM ASIC block_8x850 PIN INT* CLKNET clk100 9 ns;
```

```
MULTICYCLE TO CELL "data_out*" 2 X;
MULTICYCLE FROM CELL "compare*" 5 X ;
```

OUTPUT LOAD

Used to modify loading on output buffers from the default 0 pFs for timing analysis.

Note

This preferences is valid for all FPGAs. However, it applies only to certain I/O buffer types, generally xxCMOS and xxTTL, and to those that are only supported through hardware data within each device family. Refer to technical support.

OUTPUT LOAD is not interpreted by the Diamond Power Calculator for power estimation. Power Calculator allows you to model the average capacitance loading on outputs using the Cload parameter (default 5 pF). See the "I/O" on page 765 section of the Power Calculator online Help for more information.

Device Support All

Syntax OUTPUT [ALLPORTS | PORT <*port_name*> | GROUP <*group_name*>]
LOAD <*capacitance*> ;

where:

<*capacitance*> ::= <*number*> [mF | uF | nF | pF | fF] (Refer to the Units section of "Preference Syntax Guidelines and Conventions" on page 527.)

Note

The use of wildcards is supported for port name identifiers.

PERIOD

Identifies a clock period for all sequential output to sequential input pins clocked by the specified net or port. If no net or port name is given, the specified clock period will apply to all sequential input pins that do not have a specific clock period preference. The period preference allows you to verify sequential data paths internal to the design. It works the same way as the frequency preference, except that time is specified in seconds instead of Hz.

Device Support All

Syntax PERIOD <*time_spec*> [[LOW | HIGH] <*time_spec*> [(LOW | HIGH) <*time_spec*>]] [HOLD_MARGIN <*time_spec*>] [CLOCK_JITTER <*time_spec*>];

PERIOD NET <*net_name*> <*time_spec*> [[LOW | HIGH] <*time_spec*> [(LOW | HIGH) <*time_spec*>]] [PAR_ADJ <*par_adj_number*>] [HOLD_MARGIN <*time_spec*>];

PERIOD PORT <*port_name*> <*time_spec*> [<*polarity*> <*time_spec*> [<*polarity*> <*time_spec*>]] [PAR_ADJ

<*par_adj_number*>] [HOLD_MARGIN <*time_spec*>] [CLOCK_JITTER <*time_spec*>];

where:

<*time_spec*> ::= <*number*> <*time_units*>; (defaults to MHz)

<*polarity*> ::= LOW | HIGH

LOW | HIGH <*time_spec*> specifies pulse width and polarity, effectively allowing you to specify high/low duty cycle and phase.

<*par_adj_number*> ::= Float value for the PAR preference adjustment. This is an absolute value used to adjust the frequency. Defaults to 0.0.

The PAR_ADJ keyword allows you to tighten requirements for placement and routing while preserving the requirements reported by the TRACE static timing analysis tool. The variance in required timing values used in PAR and TRACE allows you to experiment more efficiently with over-constraining your design to determine your best constraint settings for achieving the desired results.

Note

In Diamond, the timing score reported in the PAR report is also based on the actual constraint and not on the PAR_ADJ value.

Constraints will always be tightened when PAR_ADJ is used, and only positive values can by used for this keyword. You can only use absolute values, not percentages. See the examples below for acceptable usage.

The HOLD_MARGIN keyword allows you to specify a hold margin value per clock domain to avoid possible hold time violations based on design constraints. A hold margin is an allowance of additional hold time for a given path that might be close to failing, as evidenced by timing analysis. It allows you to over-constrain the design. When this keyword is present, TRACE will read the user-supplied hold time preferences and apply them to the timing analysis. TRACE will report any path as in violation that fails to meet both the path constraint and the required hold time margin. The hold margin time defaults to 0.0.

For example, TRACE reports a given path with a 1 ps hold margin. It is very likely that a given hold time path that passes at 1 ps will fail on the actual silicon, so a margin is necessary. Prior versions of TRACE would automatically add in a guard band for hold times; however, with the current built-in min/max timing, you should set this margin manually. Generally, adding hold time margins is done with a fixed value such as 200 ps per clock domain. Some clock domains might require less or more of a margin. Hold time margins cannot be achieved by adjusting the fMAX.

The CLOCK_JITTER keyword allows you to specify a peak-to-peak jitter value for the incoming clock and the system jitter affecting the clock port. CLOCK_JITTER is only available for clock ports.

Using Wildcards For the logical PERIOD preference, wildcard expressions are allowed for NETs and PORTs. See "Using Wildcard Expressions in Preferences" on page 533 for details about wildcard usage and range expansion.

Examples The following example assigns a clock period of 100 nanoseconds to the net named NetA, with the first pulse being high with a duration of 75 ns:

```
PERIOD NET "NetA" 100 NS HIGH 75 NS;
```

The following example assigns a clock period of 30 nanoseconds to the port named Clk1:

```
PERIOD PORT "Clk1" 30 NS;
```

The following example assigns an adjusted PAR value so that PAR and TRACE timing requirements vary and create desired results for place-and-route and timing reporting. The preference keyword PAR_ADJ in the following preference uses the simple subtraction formula of 5 - 0.5 = 4.5 ns to direct PAR, while TRACE is running on the 5 ns value:

```
PERIOD NET "CLK2_c" 5 ns PAR_ADJ 0.5 ;
```

The following preferences show how the adjusted PAR value interacts with the polarity (HIGH/LOW) parameters. The two example lines both indicate a waveform of 60 percent duty cycle: 100 PERIOD with 60 HIGH, and 90 PERIOD with 54 HIGH. The PAR_ADJ in the first preference causes PAR to reduce the PERIOD by 10 ns, or 10 percent. Similarly, PAR will then reduce the HIGH time by 10 percent. Therefore the two preferences are equivalent in PAR. TRACE (**trce**) will still treat the first preference as 100 ns / 60 ns.

```
PERIOD NET  "CLK3" 100 ns  HIGH 60 ns  PAR_ADJ 10;
PERIOD NET  "CLK3" 90 ns  HIGH 54 ns
```

The following example specifies a hold margin value for each clock domain:

```
PERIOD NET "RX_CLKA_CMOS_c" 10.000 ns HOLD_MARGIN 1 ns;
```

This preference directs that any data path must meet the hold time requirement by at least 1 ns. Those paths that meet the hold time by less than 1 ns will fail in the TRACE report.

The following example assigns a clock period of 30 ns to a port named clk1 and a clock jitter of 1 ns. The jitter will affect both the rising edge and falling edge timing analysis.

```
PERIOD PORT "clk1" 30 ns CLOCK_JITTER 1 ns ;
```

See Also ▶"SYSTEM_JITTER" on page 1260

PRIORITIZE

Assigns a weighted importance to a net or bus. Values range from 0 (lowest priority) to 100 (highest). The default is 3. Any net with a priority of 3 is not considered critical and will not cause a preference to be generated. The PRIORITIZE preference is used by PAR, which assigns long lines by net priority and routes higher-priority nets before routing lower-priority nets. The PRIORITIZE preference is also used by BITGEN to determine which nets not to use for tiedown. A net with a priority greater than 3 will only be used for tiedown as a last resort.

Device Support All

Syntax PRIORITIZE NET <*net_name*> ;

PRIORITIZE BUS <*bus_name*> 0-100;

Example The following command assigns a priority of 10 to a net named NetB:

```
PRIORITIZE NET "NetB" 10;
```

This change will merely give the specified net "first priority" over clock nets with lower priorities. It will not guarantee that the net will be awarded a clock spine. The preference will resolve ordering of clock nets, but will NOT force a net to get a clock spine if doing so would dislodge a tristate bus.

PROHIBIT

Prohibits the use of a site within PAR.

Device Support All

Syntax PROHIBIT SITE <*site_name*>;

PROHIBIT REGION <*region_name*>;

Examples The following command prohibits a site named AB from being used:

```
PROHIBIT SITE "AB";
```

Note

The term FF is a keyword. Any reference to a design element named "FF" must be placed within quotation marks. For example, PROHIBIT SITE "FF";

The following command prohibits a region named reg1 from being defined in the design:

```
REGION "reg1" R2C3 4 5;
PROHIBIT REGION "reg1";
```

PROHIBIT EDGE

Prohibits the use of edge spine resources for routing the net. The edge spine can drive clocks, CEs, resets, or any combination. This preference can only be used for devices with edge clock routing in the architecture.

Device Support LatticeECP/2M

Syntax PROHIBIT EDGE NET <*net_name*>;

where:

<*net_name*> = string

Example This command prohibits the use of an edge clock to route a clock net named edge_clk.

```
PROHIBIT EDGE NET “edge_net”;
```

PROHIBIT PRIMARY

Prohibits the use of primary clock resources for routing the net. The specified net should be a clock net. This preference can be used with all devices that have primary clocking structures.

Device Support All

Syntax PROHIBIT PRIMARY NET <*net_name*>;

where:

<*net_name*> ::= string

Example This command prohibits the use of a primary clock in routing a clock net named bf_clk.

```
PROHIBIT PRIMARY NET "bf_clk";
```

PROHIBIT SECONDARY

Prohibits the use of secondary spine resources for routing the net. The secondary spine can drive clocks, CEs, resets, or any combination. This preference can be used with all devices that have secondary clocking structures.

Device Support All

Syntax PROHIBIT SECONDARY NET <*net_name*>;

where:

<*net_name*> ::= string

Example The following command prohibits the use of a secondary clock to route a clock net named bf_clk:

```
PROHIBIT SECONDARY NET "bf_clk";
```

REGION

Most commonly used to define a rectangular area that will be used by one or more universal groups (UGROUPs). See “UGROUP” on page 1263. UGROUPs explicitly define what components are members of the group. When used with the PROHIBIT REGION preference syntax, the REGION preference excludes an area from placement. These types of REGIONs are always bounded and anchored to a PFU, EBR, or DSP site, and they can overlap.

For LatticeECP3 devices, the REGION preference can also be used to define a secondary clock resource area. After the clock REGION is defined, you can use the USE SECONDARY preference to assign secondary routing. Only a clock REGION can be used for this purpose. See “Clock REGION Syntax” on page 1245 below. Refer to the device’s data sheet for more information about clock regions.

Device Support All

Device Support for Clock REGION LatticeECP3

Notes

For certain MachXO2 devices, the first EBR site on the left is positioned at column 1 and cannot be included in an anchored group or region that contains SLICEs. The column 1 EBR site is not aligned with SLICEs, which begin at column 2. Affected devices include LCMXO2-1200/4000/7000/10000.

REGION Syntax REGION <*region_name*> <*general_space*> <*site_name*> <*height*> <*width*> [DEVSIZE | PFUSIZE];
[PROHIBIT REGION <*region_name*>]

where:

<*region_name*> is a user-defined name of the REGION

<*general_space*> ::= <*site_name*> <*height*> <*width*>

<*site_name*> is a row/column Slice D location of the target device in the format R<*n*>C<*m*>D, an EBR site in the format EBR_R<*n*>C<*m*>, or a DSP site in the format DSP_R<*n*>C<*m*>.

<*height*> <*width*> is the height and width of the region in rows and columns, the two coordinates of a site address.

[DEVSIZE | PFUSIZE] is optionally used to indicate that the REGION height accounts for non-PFU rows. PFUSIZE indicates that all rows are PFU rows. The number of PFUs available to a REGION will depend on where it is anchored with the <*site_name*> parameter. DEVSIZE mode is the default behavior if the keyword is not specified.

Note

In Diamond and in ispLEVER 7.0 or later, the PFUSIZE dimension mode has been demoted and will be flagged as a syntax error during design mapping.

[PROHIBIT REGION <*region_name*>] indicates that the area should be excluded during the placement phase of automatic place and route.

Clock REGION Syntax A clock REGION must be defined with <region_name> <region_clock_space>. This can be done manually in the .lpf file, but it is usually done in Diamond's preference tools for the purpose of assigning secondary routing. When used, the REGION <region_name> <region_clock_space> preference must precede the USE SECONDARY preference that references it in the logical preference file.

REGION <*region_name*><*region_clock_space*>

where:

<*region_name*> is a user-defined name of the clock REGION

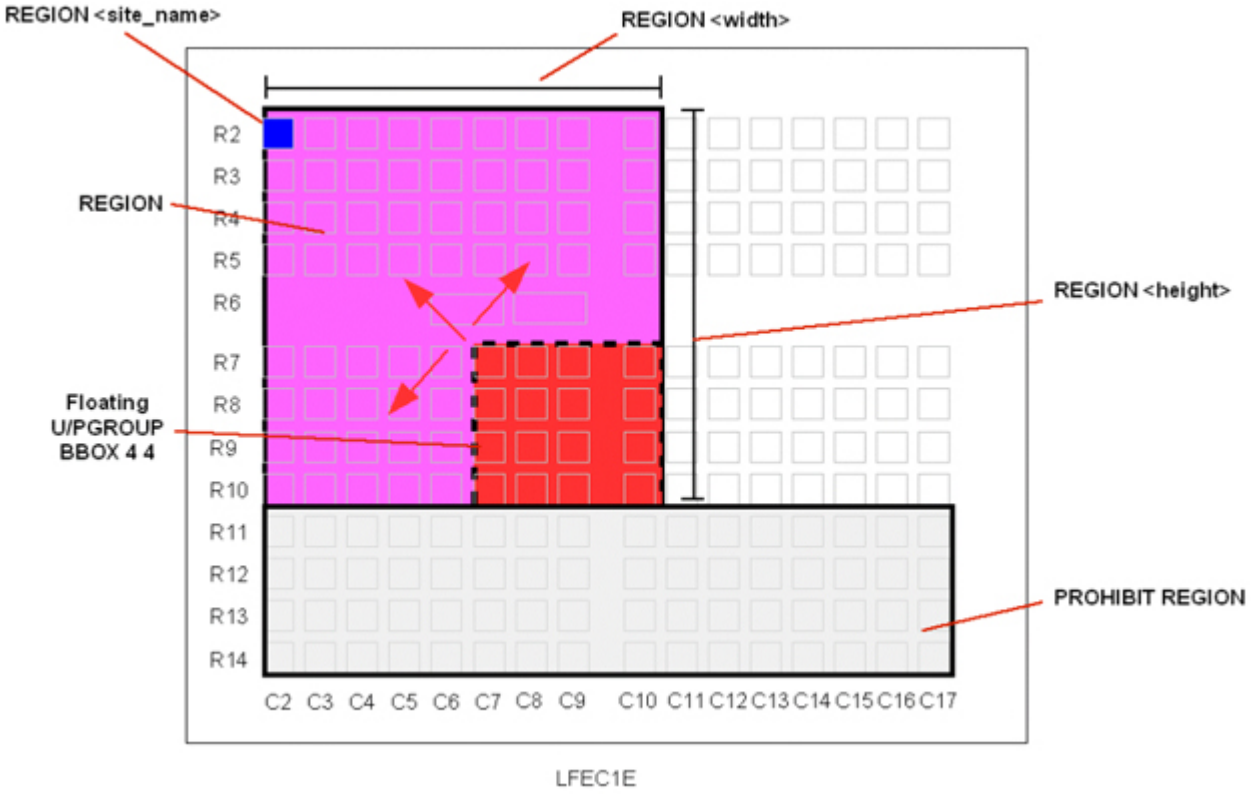
<*region_clock_space*> ::= CLKREG <*clock_region_name*> [<*cr_height*> <*cr_width*>];

<*clk_region_name*> is the site name that identifies the clock_region; for example, CLKREG_R2C2

<*cr_height*><*cr_width*> are integers that describe how many clock_regions are in the region_clock_space. If these are omitted, the default value will be 1, thus creating a region_clock_space of a single clock_region.

Usage REGION is usually defined from within Diamond's Floorplan View, where an abstract view of the target device sites is presented. You can also specify a REGION directly within the .lpf file by specifying a logical preference. REGION can be added directly to HDL code to associate a UGROUP attribute with a REGION. A REGION can also be declared as part of a VHDL component or a Verilog module instance that is also tagged with a UGROUP attribute.

The figure below illustrates two REGION definitions super-imposed on an LFEC1E device floorplan.



Example 1 – Floating UGROUP within a REGION The REGION, UGROUP, and LOCATE UGROUP preference enables you to define a placement group that will float within a certain region. The UGROUP can be bounded or not:

```
REGION "Region_0" "R16C2D" 11 30 DEVSIZE;
UGROUP "mux_group" BBOX 4 4
  BLKNAME mux_1;
LOCATE UGROUP "mux_group" REGION "Region_0" ;
```

Example 2 – Reserved REGION The REGION and PROHIBIT preference enables you to define a region where no logic will be placed by Place & Route:

```
REGION "Region_0" "R16C2D" 11 30 DEVSIZE;
PROHIBIT REGION "Region_0" ;
```

Example 3 – LPF-based Clock Region This REGION preference indicates a 2 x 3 clock region anchored in the upper left corner by clock region CLKREG_R2C2. The REGION will be two clock_regions high and three clock_regions wide. The USE SECONDARY NET preference indicates that all loads related to clk1 are to be placed in the area defined by region_mux.

```
REGION “region_mux” CLKREG “CLKREG_R2C2” 2 3;
USE SECONDARY NET “clk1” REGION “region_mux”;
```

RVL_ALIAS

Provides an alias name for a clock name generated by Reveal Inserter so that it can be mapped to the original clock name used in the design. This ensures that any preferences that use the Reveal clock names will be treated as if they were using the original clock names.

Device Support All

Note

The RVL_ALIAS preference is automatically generated and should not be added manually or edited.

Syntax RVL_ALIAS <reveal_clock_name> <original_clock_name>;

Example `RVL_ALIAS "reveal_ist_9" "clock1/clk_sample";`

SSO

Defines constraints for Simultaneous switching output (SSO) I/O analysis.

Device Support All

Usage The most common usage of SSO constraints is to define them from within Spreadsheet View. You can also specify SSO parameters directly within the .lpf file by specifying the SSO logical preference. SSO I/O analysis reports can be generated from Spreadsheet View.

Syntax SSO [ALLPORTS] (<*keyword*>=<*value*>)+;
SSO [PORT <*port_name*> | GROUP <*group_name*>] (<*keyword*>=<*value*>)+

where:

<*port_name*> = string. These are not the actual top-level port names but the signal name attached to the port. Any multiple listings or wildcards should be done using GROUPS.

<*group_name*> = string

<*keyword*> (Output/Bidir) ::= SwitchingID, SSO_Noise, SSO_Noise_Drop, SSO_Allowance

SwitchingID is an integer identifier to group related pins that will switch at the same time.

SSO_Noise is the estimated noise on the PCB board's ground plane, expressed as a floating point value for millivolt units (mV). The default is 0.00 mV.

SSO_Noise_Drop is the estimated noise on the PBC board's power plane for drop calculations, expressed as a negative floating point value for millivolt units (mV). The default value is -0.00 mV.

SSO_Allowance is the allowable level of I/O SSO per bank expressed as a percentage (%). The default is 100%.

For the SSO ALLPORTS preference, any possible subordinate attributes of SSO, such as SwitchingID will take effect only on ports that have no other specific SSO preferences placed upon them. This is because of the preference precedence rule: more specific preferences override more generally applied ones.

Using Wildcards The use of wildcards for SSO is supported for port names.

```
SSO ALLPORTS SwitchingID=0 SSO_Noise=0.00 SSO_Noise_Drop=-10.00
SSO_Allowance=70;

SSO PORT "port1" SwitchingID=1 SSO_Noise=20.00
SSO_Noise_Drop=-10.00 SSO_Allowance=80;

SSO PORT "port2" SwitchingID=1 SSO_Noise=20.00
SSO_Noise_Drop=-10.00 SSO_Allowance=80;

SSO PORT "port3" SwitchingID=1 SSO_Noise=20.00
SSO_Noise_Drop=-10.00 SSO_Allowance=80;

SSO PORT "port4" SwitchingID=1 SSO_Noise=20.00
SSO_Noise_Drop=-10.00 SSO_Allowance=80;
```

SYSCONFIG

Defines system configuration option settings for the sysCONFIG feature. If you do not specify these settings in the .lpf file, using the Global Preference sheet in Spreadsheet View or manually, some default sysCONFIG preferences will automatically be generated based on device selection.

The SYSCONG preference is available for all Lattice FPGA devices that support the sysCONFIG configuration port. The sysCONFIG port can be a single data line or byte-wide (multiple byte) data port that can support serial and parallel `configuration` streams. The devices also support daisy chaining.

Device Support All

Syntax SYSCONFIG <*keyword*>=<*value*>+

where:

<*keyword*> refers to any of the following keywords. Click on a keyword for further description.

<*value*> refers to any possible values associated with that keyword. The allowed values for the above keywords are device specific.

Recommended Usage You should set the preference configuration options in the pre-map stage using Spreadsheet View or by editing the .lpf file directly. Also see "Setting Global Preferences" on page 461 for details. If you do not set sysCONFIG options, default SYSCONFIG preferences are automatically set in the .prf file based on device selection.

Examples The SYSCONFIG preference allows you to set a series of keyword values in succession after the preference identifier.

```
SYSCONFIG CONFIG_MODE=SLAVE_SERIAL PERSISTENT=OFF DONE_OD=ON;

SYSCONFIG PERSISTENT=OFF CONFIG_MODE=SLAVE_SERIAL WAKE_UP=21
MCCLK_FRQ=0.78 COMPRESS_CONFIG=ON;
```

These examples would set the configuration mode to serial slave. The ability to retain system configuration pin reservation, even during normal operation, is turned OFF with the PERSISTENT keyword. The DONE pin will be configured as an open drain pin when the DONE_OD keyword set to ON.

SYSCONFIG Keyword Settings The following table shows the default settings in bold type and the selectable settings for all of the keyword values for the SYSCONFIG preference. For more details on keyword values pertaining to device requirements, refer to the usage guides under See Also at the end of this topic.

Table 139:

Keyword	Default and Selectable Values	Supported Devices
BACKGROUND_RECONFIG	**OFF** [off, on, sram_only, sram_ebr]	ECP5U/UM, MachXO2, MachXO3D, MachXO3L, MachXO3D
BACKGROUND_RECONFIG_SECURITY	**OFF** [off, on]	MachXO3D
COMPRESS_CONFIG	**ON** [on, off]	LatticeECP/EC, LatticeECP2/M, Lattice ECP3, ECP5, LatticeSC/M, MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
CONFIG_IOVOLTAGE	**2.5** [2.5, 1.2, 1.5, 1.8, 3.3]	LatticeECP3, ECP5
CONFIG_MODE	Available values vary, depending on the device. Refer to the usage guides under See Also at the end of this topic.	All except MachXO, MachXO2, MachXO3D
CONFIG_SECURE	**OFF** [off, on]	All
CONFIGURATION	CFG [CFG, CFG_EBRUFM, CFGUFM, EXTERNAL]	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
CUR_DESIGN_BOOT_LOCATION	**IMAGE_0** [IMAGE_0, IMAGE_1]	MachXO3D
CUSTOM_IDCODE	32-bit arbitrary	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
CUSTOM_IDCODE_FORMAT	**Binary** [Binary, Hex]	MachXO3D
DONE_EX	**OFF** [off, on]	All except MachXO2, MachXO3D
DONE_OD	**ON** [off, on]	LatticeECP/EC, LatticeXP/2, LatticeECP2/M, LatticeECP3, ECP5
DONE_PULL	**ON** [off, on]	LatticeSC/M, ECP5
DUALBOOTGOLDEN	**INTERNAL** [internal, external]	MachXO3LF
ENABLE_NDR	**OFF** [off, on]	LatticeECP2/M, LatticeECP3
ENABLE_TRANSFR	**DISABLE** [disable, enable]	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2

Table 139:

Keyword	Default and Selectable Values	Supported Devices
I2C_PORT	**DISABLE** [disable, enable]	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
INBUF	**OFF** [off, on] for ECP5, LatticeECP/EC, LatticeECP2/M **ON** [off, on] for LatticeXP/2and MachXO, Platform Manager	ECP5, LatticeECP/EC, LatticeECP2/M, LatticeXP/2, MachXO, Platform Manager
JTAG_PORT	**ENABLE** [enable, disable]	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
MASTER_PREAMBLE_DETECTION_RETRY	**0** [0 to 3]	MachXO3D
MASTER_PREAMBLE_DETECTION_TIMER	**0** [0 to 15]	MachXO3D
MASTER_SPI_PORT	**DISABLE** [disable, enable]	ECP5, MachXO2, MachXO3D, MachXO3L, Platform Manager 2, LatticeXP2, not available in MachXO3D
MCCLK_FREQ	Lowest Frequency. Refer to the usage guides under See Also at the end of this topic.	All except LatticeXP2
MUX_CONFIGURATION_PORTS	**DISABLE** [disable, enable]	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
MY_ASSP	**OFF** [off, on]	LatticeXP2, MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
ONE_TIME_PROGRAM	**OFF** [off, on]	LatticeXP2, MachXO2, MachXO3D, MachXO3L, Platform Manager 2, not available in MachXO3D
PERSISTENT	**OFF** [off, on] exclusive of LatticeECP3. **OFF** [off, SLAVE_PARALLEL, SSPI] for LatticeECP3 devices.	LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeSC/M
PRIMARY_BOOT	**IMAGE_0** [image_0, image_1, external, latest, former]	MachXO3D
	DISABLE [disable, enable]	MachXO3D
SDM_PORT	**DISABLE** [disable, programn, programn_done, programn_done_initn, initn, done]	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
SECONDARY_BOOT	**NONE** [none, image_0, image_1, external, latest, former]	MachXO3D

Table 139:

Keyword	Default and Selectable Values	Supported Devices
SFDP_CHECK	**DISABLE** [disable, enable_sfdp, enable_sfdp_preamble]	MachXO3D
SHAREDEBRINIT	**DISABLE** [disable, enable]	MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2
SLAVE_IDLE_TIMER	**0** [0 to 15]	MachXO3D
SLAVE_PARALLEL_PORT	**DISABLE** [disable, enable]	ECP5
SLAVE_SPI_PORT	**DISABLE** [disable, enable]	ECP5, LatticeXP2, MachXO2, MachXO3D, MachXO3L, MachXO3D
SPIM_ADDRESS_32BIT	**DISABLE** [disable, enable]	MachXO3D
STRTUP	**EXTERNAL** [external, TCLK, CCLK, MCLK]	LatticeECP3
TRANSFR	**OFF** [off, on]	ECP5
WAKE_ON_LOCK	**OFF** [off, on]	LatticeXP2, LatticeECP3
WAKE_UP	**21** (DONE_EX = Off) [1:25] For LatticeECP3, selectable values are 1, 4, 6, 7, 10, 14, 17, 21, 22, 23, 24, 25. **4** (DONE_EX = On) [1:7]	All except MachXO2, MachXO3D

BACKGROUND_RECONFIG= OFF (default) | ON [once] | sram_only | sram_ebr

Allows user to reconfigure device in user mode. The use of this preference is risky and is recommended for advanced users only.

BACKGROUND_RECONFIG_SECURITY=OFF (default) | ON

Allows user to control the CFG access to the ESB. When set to OFF, CFG has read only access. When set to ON, CFG can read/write to ESB.

BULK_ERASE=YES (default) | NO

Allows user to set the bulk erase. When set NO, auto bulk erase is disabled. When set to YES, auto SRAM bulk erase is allowed for power cycling, refresh, or PROGRAMN pin toggling.

COMPRESS_CONFIG=OFF (default) | ON [once]

Bitstream Compression. When set to ON, for those devices that support bitstream compression, the software generates a compressed version of the bitstream file.

CONFIG_IOVOLTAGE=2.5 (default) | 1.2 | 1.5 | 1.8 | 3.3

Sets the voltage for the sysCONFIG bank. When you set this attribute, it tells the software the voltage that is required in this bank to satisfy the sysCONFIG requirements. DRC errors will then be based on CONFIG_IOVOLTAGE and usage of the dual-purpose sysCONFIG pins.

CONFIG_MODE= mode_name

The CONFIG_MODE option tells the software to preserve the appropriate sysCONFIG ports/pins to help the user design flow. After you determine what persistent sysCONFIG pins are available for mapping, the resulting CONFIG_MODE preference will tell the software what you intend to do with the sysCONFIG modes.

The available CONFIG_MODE options sysCONFIG modes are device-specific because each device supports a different set of ports. Refer to the usage guides under See Also at the end of this topic for more information.

The PERSISTENT keyword, when set to ON (for devices other than LatticeECP3), will communicate to the software that the selected CONFIG_MODE attribute is to be PERSISTENT. See the description of PERSISTENT below for more information.

CONFIG_SECURE= OFF (default) | ON [once]

Configuration Security. When set to ON, no readback operation is supported through the sysCONFIG port or ispJTAG port of the general contents. The USERCODE area is readable and not considered securable. The OFF setting indicates that readback is enabled through any port. LatticeSC supports an additional selection ONCE, which blocks the readback operation once.

CONFIGURATION=CFG (default) | CFG_EBRUFM | CFGUFM | EXTERNAL

Configuration settings for storing the configuration bitstream for MachXO2, MachXO3D, MachXO3L, and Platform Manager 2 devices. When set to CFG, the configuration bitstream (including EBR init if any) is stored in the configuration Flash array and is not allowed to overflow into the UFM.

When set to CFG_EBRUFM, the configuration bitstream (NOT including EBR init) is stored in the configuration Flash array. EBR init data, if any, is stored in the lowest page addresses (starting from Page 0) of the user Flash memory (UFM). Any unoccupied UFM pages after the mapping of EBR init data, will be available as general purpose Flash memory.

When set to CFGUFM, the configuration bitstream (including EBR init if any) is allowed to overflow into the UFM.

When set to EXTERNAL, the configuration bitstream (including EBR init if any) is stored in an external memory device. Use this setting when the bitstream does not fit in the internal flash resources of the device. The MASTER_SPI_PORT preference must be set to ENABLE to enable the external Master SPI port for configuring the SRAM fuses using the bitstream.

CUR_DESIGN_BOOT_LOCATION=IMAGE_0 (default) | IMAGE_1

The value (IMAGE_0 and IMAGE_1) determines the .jed filename. For IMAGE_0, <project_name>_a.jed is generated. For IMAGE_1, <project_name>_b.jed is generated. Either project can be downloaded into either FLASH-A or FLASH-B.

CUSTOM_IDCODE=32-bit arbitrary value

This attribute is a 32-bit arbitrary value. When you enable the MY_ASSP feature, this field becomes the JTAG ID of the device.

CUSTOM_IDCODE_FORMAT=Binary (default) | Hex

The user specifies the format for the CUSTOM_ID. The available formats are Hex and Binary.

DONE_EX= OFF (default) | ON

External DONE Control. Determines whether the device wake-up sequence is synchronized with its own internal DONE signal or with an external DONE signal driven by another device. If ON, the device will wait for an external DONE signal to be driven high to wake up. In its OFF default state, the device wakes up on its own after the DONE bit is set.

Specify ON if you want to delay wake up until the DONE pin is driven high by an external signal and synchronous to the clock. Specify OFF if you wish to synchronously wake up when the DONE bit is set and ignore any external driving of the DONE pin. If DONE_EX is set to ON, DONE_OD should be set to its default value of ON. If an external signal is driving the DONE pin, it should be an open drain pin. See DONE_OD for information on DONE pin sysCONFIG and active versus open drain concepts.

DONE_OD= ON (default) | OFF

Done Pin Open Drain. Enables you to configure the DONE pin as an open drain pin. By default, the pullup on the DONE pin is active.

When the DONE pin is released high, this indicates that programming is complete and the device is ready for wake up. Other devices can be used to control the wake-up process of the device by holding the DONE pin low until they are ready to wake up. To allow other devices to hold the DONE pin low, an open drain configuration is needed to avoid contention on the DONE pin from the device trying to drive high when other DONE signals are trying to hold it low.

DONE_PULL= ON (default) | OFF

Internal DONE Pin Pullup. The DONE_PULL keyword enables you to configure the DONE pin with a pullup on or off. The DONE_PULL keyword is only used for the DONE pin. When the DONE pin is driven low, this indicates that programming is complete and the device is ready for wakeup. Other devices can be used to control the wake-up process of the device by holding the DONE pin low. This allows the pin to pull itself high when other devices have released the open drain DONE pin. When you do not want to use an internal active pullup, set the DONE_PULL keyword to OFF.

DUALBOOTGOLDEN= INTERNAL (default) | EXTERNAL

MachXO3D and MachXO3L/LF devices can optionally boot from two patterns, a primary bitstream and a golden bitstream. If the primary bitstream is found to be corrupt while being downloaded into the SRAM, the device shall then automatically re-boot from the golden bitstream. The DUALBOOTGOLDEN preference defines if the Golden file is Internal (on-chip Flash/NVCM) or External (SPI Flash).

ENABLE_NDR=OFF (default) | ON

Non Disruptive Reconfiguration (NDR). When turned on, allows volatile devices to be reconfigured while I/Os are in Leave Alone I/O mode. This is done by latching all user I/Os (except sysCONFIG pins) to a known state when the control register's NDR control bit (bit 25 of control Register 0 for LatticeECP2) is set while the PROGRAMN pin is toggled or when a JTAG refresh command is issued. When turned off, the I/Os are tri-stated during reconfiguration.

ENABLE_TRANSFR= DISABLE (default) | ENABLE

When enabled, turns on the Freeze_IO and Freeze_MIB fuses to enable the TransFR function. The Freeze_IO holds the current logic value (data and tristate) on the pins constant. The Freeze_MIB holds the personalities of the pins constant. When this is enabled, the IOs will not be changed in terms of logic value and all buffer modes of the IOs.

I2C_PORT=DISABLE | ENABLE (default)

For MachXO2, MachXO3D, MachXO3L, and Platform Manager 2 devices, enabling the I2C_PORT makes it active for configuration access and prevents the dual-purpose pins from being used as user I/O pins. When disabled, the pins become available for user I/Os.

INBUF=ON | OFF

Defaults to OFF for ECP5, LatticeECP/EC, and LatticeECP2/M devices. Defaults to ON for LatticeXP/2, MachXO, and Platform Manager devices.

Turns on the input buffers. This attribute must be turned on to perform boundary scan testing. When turned off, all unused input buffers are disabled to save power.

JTAG_PORT=ENABLE (default) | DISABLE

When enabled, it prevents the JTAG pins in a MachXO2, MachXO3D, MachXO3L, and Platform Manager device from being used as I/O pins.

When disabled, signals can be assigned to the shared JTAG pins. A high signal on the JTAGENB would then be required to re-enable the JTAG port, after which the 4 pins would become dedicated JTAG pins.

MASTER_PREAMBLE_DETECTION_RETRY=0 (default) to 3

This attribute sets the Master mode Bitstream preamble detection retry time. The attribute value is 0 to 3. The number of times the preamble detection should be retried in Master mode (booting from external Flash.)

When set to 0, preable detection does not try, then have option to try one time, two times, or three times.

MASTER_PREAMBLE_DETECTION_TIMER=0 (default) to 15

This attribute sets the Master mode Bitstream preamble detection timer count value. When set to 0, the timer's value is set to ~126Kus; there are 16 steps from 126Kus to 3.85us.

MASTER_SPI_PORT=DISABLE (default) | ENABLE | EFB_USER (MachXO2 only)

When enabled, it allows the use of the external Master SPI port for configuring the SRAM fuses using the bitstream.

When EFB_USER is selected for a MachXO2, MachXO3D, MachXO3L device, it reserves the SPI pins for user Master SPI mode in the embedded function block (EFB).

MCCLK_FREQ= <*frequency_value*> (MHz)

Controls the Master Clock frequency. Valid options are 2.5 MHz, 4.3 MHz, 5.4 MHz, 6.9 MHz, 8.1 MHz, 9.2 MHz, 10 MHz, 13 MHz, 15 MHz, 20 MHz, 26 MHz, 30 MHz, 34 MHz, 41 MHz, 45 MHz, 51 MHz, 55 MHz, 60 MHz, 130 MHz. The default is the lowest frequency, 2.5 MHz. Note that 51 MHz is not a valid option for LatticeECP2/M. Refer to the Global Preferences topic for further description.

MUX_CONFIGURATION_PORTS= DISABLE (default) | ENABLE (MachXO2 only)

Allows all Configuration ports to be disabled in order to provide additional user IOs.

MY_ASSP=OFF (default) | ON

LatticeXP2 devices – when set to ON, causes the USERCODE to be used as a custom device ID. When JTAG scanning is done, the device will produce the specified device ID pre-programmed onto the USERCODE fuses.

MachXO2, MachXO3D, and MachXO3L devices – when set to ON, changes the JTAG IDCode of the device to the contents of the CUSTOM_IDCODE field.

ONE_TIME_PROGRAM=OFF (default) | ON

When set to ON, uses the One Time Programmable (OTP) fuse to prevent the on-chip Flash configuration memory from being erased or programmed. This does not prevent the Flash Tag Memory or Flash User memory from being programmed; these features are still available.

PERSISTENT= OFF (default) | ON (exclusive of LatticeECP3) | SLAVE_PARALLEL (LatticeECP3 only) | SSPI (LatticeECP3 only).

Some of the sysCONFIG pins support dual functions, meaning that they serve as sysCONFIG pins during configuration operations and I/O pins during normal operation. The PERSISTENT preference determines whether these

pins remain sysCONFIG pins during normal operation. The PERSISTENT keyword can be applied to only certain configuration modes.

When set to ON, for devices other than LatticeECP3, it reserves the dual-purpose pins for configuration and informs the software that these pins are not available as general purpose I/O pins. Note that setting a value of 'ON' can be done in the Diamond Spreadsheet View, Global Preferences tab, by using the ENABLE setting for the appropriate port.

When set to SLAVE_PARALLEL (for LatticeECP3), it signals to the software that all the dual purpose sysCONFIG pins will NOT be available for placement and routing, regardless of the selected CONFIG_MODE.

When set to SSPI (for LatticeECP3), it specifies that only the slave SPI port is to be preserved after configuration.

PRIMARY_BOOT=IMAGE_0 (default) | IMAGE_1 | EXTERNAL | LATEST | FORMER

This attribute specifies the bitstream boot location. IMAGE_0 is loaded from CFG0/UFM0 section of Flash memory, IMAGE_1 is loaded from CFG1/UFM1 section. EXTERNAL option is used for external boot, LATEST is for DUAL_INTERNAL boot - latest first, and FORMER is for DUAL_INTERNAL boot - latest second.

ROLLBACK_CONTROL=DISABLE | ENABLE

When set to ENABLE, allows the enforcement of roll back control.

SDM_PORT=DISABLE (default) | PROGRAMN | PROGRAMN_DONE | PROGRAMN_DONE_INITN | INITN | DONE

This preference is used to retain the PROGRAMN, DONE, and INITN pins as dedicated configuration pins for the self-download mode port.

SECONDARY_BOOT= NONE (default) | IMAGE_0 | IMAGE_1 | EXTERNAL | LATEST | FORMER

This attribute can be set as follows: NONE for single boot, IMAGE_0 for CFG0/UFM0, IMAGE_1 for CFG1/UFM1, EXTERNAL for external boot, LATEST for DUAL_INTERNAL boot - latest first, and FORMER for DUAL_INTERNAL boot - latest second.

SFDP_CHECK=DISABLE (default) | ENABLE_SFDP | ENABLE_SFDP_PREAMBLE

When set to DISABLE, this attribute doesn’t allow for SFDP checking. When set to ENABLE_SFDP, the SFDP checks and stops as soon as SFDP check fails. When set to ENABLE_SFDP_PREAMBLE, SFDP checks and continues PREAMBLE even though SFDP check fails.

SHAREDEBRINIT=DISABLE (default) | ENABLE

When set to ENABLE, this attribute allows sharing of the same init file that was used by multiple EBRs when they were generated in IPexpress. This reduces the bitstream size by storing only one copy of the init values.

SLAVE_IDLE_TIMER=0 (default) to 15

The attribute will set the Slave Idle Timer count value. There are 15 steps from 128Kms to 10ms; when set to 0, the Slave Idle Timer is disabled, or, in other words, set to Infinite.

SLAVE_SPI_PORT=DISABLE (default) | ENABLE

When enabled, it allows on-chip Flash to be programmed and read back using the external Slave SPI port.

SLAVE_PARALLEL_PORT=DISABLE (default) | ENABLE

When enabled, it makes the 8-bit slave parallel port available after configuration and allows the SRAM to be reconfigured.

SPIM_ADDRESS_32BIT=DISABLE (default) | ENABLE

Allows user to extend the SPIM address to 32-bit address. In default setting, SPIM is a 24-bit address. When set to ENABLE, SPIM is a 32-bit address.

STRTUP=EXTERNAL (default) | TCLK | CCLK | MCLK

Specifies the clock source to be used for synchronization with the start-up sequence and allows a user-defined clock source to be used instead of CCLK or TCK. The clock signal must be specified and the STRTUP library element instantiated in the design.

TRANSFR=OFF (default) | ON

When ON is selected, turns on the Freeze_IO and Freeze_MIB fuses to enable the TransFR function.

WAKE_ON_LOCK=OFF (default) | ON

Determines whether the device will wait for the PLL to lock before beginning the wake-up process.

By default (OFF), the device will wake up regardless of the state of the PLL lock signal. When set to ON, the device will not wake up until the PLL lock signal for the given PLL is active.

WAKE_UP= 21 (DONE_EX = OFF) | 4 (DONE_EX = ON)

Wake Up. Wake Up is a controlled event after a part has been configured. External control of the DONE pin can be selected to either delay wake up or be ignored. See DONE_EX.

The Wake Up sequence controls three internal signals, and the DONE Pin will be driven after configuration and prior to user mode. If DONE_EX = ON, the WAKE_UP keyword will take your selected option (1-7), and then the software will set wake-up signal bits accordingly. If you do not select a wake-up sequence, the default wake-up sequence will be 4. If DONE_EX = OFF (default), the WAKE_UP keyword will take your selected option (1-25), and then the software will set wake-up signal bits accordingly. If you do not select a wake-up sequence, the default wake-up sequence will be 21.

See Also ▶"Programming the FPGA" on page 802

- "Setting Global Preferences" on page 461
- TN1260, *ECP5 sysCONFIG Usage Guide*
- TN1053, *LatticeEC/ECP sysCONFIG Usage Guide*
- TN1108, *LatticeECP2/M sysCONFIG Usage Guide*
- TN1169, *LatticeECP3 sysCONFIG Usage Guide*
- TN1080, *LatticeSC sysCONFIG Usage Guide*
- TN1082, *LatticeXP sysCONFIG Usage Guide*
- TN1141, *LatticeXP2 sysCONFIG Usage Guide*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1204, *MachXO2 Programming and Configuration Usage Guide*
- TN1279, *MachXO3L Programming and Configuration Usage Guide*
- TN1303, *CrossLink Programming and Configuration Usage Guide*
- FPGA-TN-02069 - *CrossLinkFlash Programming and Config Usage Guide*

SYSTEM_JITTER

Specifies the peak-to-peak system jitter for timing analysis. This global preference is used to overwrite the default system jitter. It will affect both the rising edge and falling edge timing analysis. Trace will use half of the specified peak-to-peak jitter for all analysis.

Device Support All

Syntax SYSTEM_JITTER <*time_spec*><*time_unit*>

where:

<*time_spec*> ::= <*number*>

<*time_unit*> ::= ps | ns

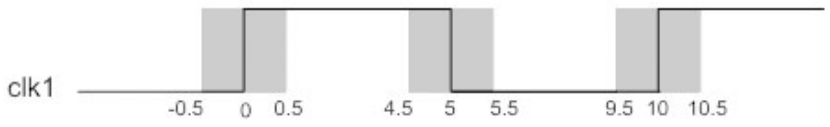
The system jitter value should not be smaller than the default system jitter for the device.

Example

```
SYSTEM_JITTER 1.0 ns;
```

In the following waveform for this example, the rising edge for clk1 is in the range of -0.5 to 0.5 nanoseconds. The falling edge is the range of 4.5 to 5.5 nanoseconds.

Figure 36:



TEMPERATURE

Assigns an operating junction temperature for the part that changes the derating factor for timing. You can use this preference to override the default temperature, which is 85C (commercial) and 100C (industrial) for all speed grades. If no temperature or voltage preference is specified, the derating factor is 1.0.

Device Support All

Syntax TEMPERATURE <*temp*>;

where:

<*temp*> ::= <*number*> (C | F | K)

Note

The temperature can be specified in C (Celsius), F (Fahrenheit) or K (Kelvin). Internally it is converted to Celsius. The use of a space between the number and the voltage symbol is optional. See examples.

Examples The following preference sets the operating junction temperature to 45 degrees Celsius.

```
TEMPERATURE 45C;
```

The following preferences set both the junction temperature and voltage derating parameters for LatticeSC/M devices.

```
TEMPERATURE 25C;
VCC_DERATE NOMINAL;
VCCAUX_DERATE NOMINAL;
VCCIO_DERATE BANK 0 NOMINAL;
VCCIO_DERATE BANK 1 NOMINAL;
VCCIO_DERATE BANK 2 NOMINAL;
VCCIO_DERATE BANK 3 NOMINAL;
VCCIO_DERATE BANK 4 NOMINAL;
VCCIO_DERATE BANK 5 NOMINAL;
VCCIO_DERATE BANK 6 NOMINAL;
VCCIO_DERATE BANK 7 NOMINAL;
```

See Also ▶"Specifying Voltage and Temperature" on page 484

- "VOLTAGE" on page 1280
- "VCC_DERATE" on page 1275
- "VCC1P2_DERATE" on page 1277
- "VCCAUX_DERATE" on page 1278
- "VCCIO_DERATE" on page 1278

TRACEID

Specifies a code for storing device data, much like USERCODE but in binary format.

Device Support MachXO2, MachXO3D, MachXO3L, MachXO3D, Platform Manager 2

Syntax TRACEID<*binary_string*>;

where:

<*binary_string*> ::= binary value of up to 8 bits

Example `TRACEID "00001111";`

TWR_REPORT_LIMIT

Specifies the number of items (paths) to be reported by the TRACE timing analysis tool (**trce**) for each preference in the .twr file.

To use this preference, the command-line options -v and -e must not be used. Otherwise, the path limit defined at the command line will override this preference.

Device Support All

Syntax TWR_REPORT_LIMIT <*number*>;

where:

<*number*> ::= integer;

Example This command specifies that TRACE reports 20 paths in the output .twr file:

```
TWR_REPORT_LIMIT  20;
```

UGROUP

Universal logical grouping constraint. UGROUP is used to infer a single placement group (PGROUP) for multiple instances of an object from the logical domain. UGROUPs are translated by the design mapper (map) into PGROUPs, which direct the placer algorithm of PAR to place the group members in proximity. UGROUP members refer to logical component names in the EDIF/NGD netlist. Universal groups can include PFU/PFF, EBR, or DSP-based logic.

The hierarchical grouping constraint, HGROUP, is an alternative to UGROUP and is used to infer a unique placement group for each instance of a block.

UGROUP is also available as an HDL attribute that can be added directly to HDL source as a means to infer PGROUPs from a VHDL architecture or component instance or from a Verilog module declaration or module instance.

Device Support All

Usage A UGROUP preference can be defined from within Spreadsheet View or Floorplan View, where the post-synthesis netlist (EDIF/NGD) is presented, or it can be defined directly within the logical preference file (.lpf). UGROUPs can be anchored or floating, bounded or unbounded. UGROUPs that include DSP blocks must be anchored. If a floating UGROUP that includes a DSP block is encountered, the system will issue a warning and ignore the block.

A UGROUP can also be defined as an HDL attribute as part of a VHDL component or Verilog module instance. When Diamond encounters a UGROUP declaration from the post-synthesis netlist, it presents it in Spreadsheet View. If you modify a UGROUP that originated in the post-synthesis netlist, Diamond will add the UGROUP declaration to the .lpf file and override the original inferred group based on the precedence rules of the Diamond preference flow.

One or more UGROUPs can be members of a REGION. Any PFU, EBR, or DSP site can serve as an anchor. UGROUPs can overlap as long as enough device sites are available to accommodate the design logic. A given block can only be assigned membership to one group, either a UGROUP or an HGROUP. See “Troubleshooting UGROUP Conflicts” on page 1265.

Members of a UGROUP are automatically translated by the design mapper into PGROUPs that comprise post-map physical domain elements. After mapping, PGROUPs inferred from UGROUPs of the post-synthesis netlist appear in the schematic section of the physical preference file (.prf), and those inferred from the .lpf file appear in the user defined section of the .prf file. This organization enables you to easily override an HDL-based declaration in the post-synthesis stage. If you wish to later restore an HDL-

based UGROUP, use the Group sheet in Spreadsheet View to delete the UGROUP from the .lpf file.

Note

The legacy PGROUP preference is no longer supported in the logical preference file for grouping PFF, PFU, and DSP-based logic. Use the UGROUP preference instead.

Syntax UGROUP <*ugroup_name*> [BBOX <*height width*>] (BLKNAME <*blkname*>)+;
[LOCATE UGROUP <*ugroup_name*> SITE <*site_name*>;]
[LOCATE UGROUP <*ugroup_name*> REGION <*region_name*>;]

where:

<*ugroup_name*> is the user-defined name of the UGROUP.

BBOX <*height_width*> is used to optionally define the maximum number of rows (R) and columns (C), of the group's bounding box. The bounding box can include PFF, PFU, EBR, and DSP sites.

Note

For certain MachXO2 devices, the first EBR site on the left is positioned at column 1 and cannot be included in an anchored group or region that contains SLICEs. The column 1 EBR site is not aligned with SLICEs, which begin at column 2. Affected devices include LCMXO2-1200/4000/7000/10000.

BLKNAME <*blkname*> Identifies the name of the logical block you are grouping. A given block can only be assigned membership in one group, either a UGROUP or an HGROUP (hierarchical group). It cannot be in two or more groups at the same time.

[LOCATE UGROUP <*ugroup_name*> SITE <*site_name*>] is used to optionally specify the UGROUP position. The <*site_name*> can be a Slice D row/column location of the target device in the format R<*n*>C<*m*>D; it can be an EBR site in the format EBR_R<*n*>C<*m*>; or it can be a DSP site in the format DSP_R<*n*>C<*m*>.

[LOCATE UGROUP <*ugroup_name*> REGION <*region_name*>] is used to optionally specify the REGION within which the UGROUP will float.

Example Given a UGROUP and LOCATE preference:

```
UGROUP "rot_grp" BBOX 8 3
  BLKNAME rotate_1;
LOCATE UGROUP "rot_grp" SITE "R2C23D"
```

After design mapping, the resulting .prf file will contain the following in the schematic section:

```
PGROUP "rot_grp" BBOX 8 3  DEVSIZE
  COMP "rotate_1/SLICE_0"
  COMP "rotate_1/SLICE_1"
  COMP "rotate_1/SLICE_2"
  COMP "rotate_1/SLICE_3";
LOCATE PGROUP "rot_grp" SITE "R2C23D" ;
```

Troubleshooting UGROUP Conflicts Multiple definitions of the same group name can sometimes appear in the .lpf file because of changes in the design or semantic errors. For example, a UGROUP in the .lpf ASCII file might contain a semantic error and not show up in the Diamond preference tools. When the UGROUP is recreated in the preference tools, it produces redundant groups in the .lpf file.

When the software encounters a UGROUP conflict in the .lpf file, it produces warnings messages and processes the groups using the following rules:

- Only one group definition is recognized for each UGROUP name. When two group definitions with the same name appear in the .lpf file, only the UGROUP listed first is recognized.
- UGROUPs cannot share the same members. When two UGROUPs with different names include some of the same elements, only non-conflicting elements are assigned to each group. When each group contains identical members, the first group is ignored.

See Also ▶"HGROUP" on page 1220 (preference)

- "UGROUP" on page 1329 (HDL attribute)

UNIQUE_ID

Defines the upper sixteen bits of an automatic 32-bit USERCODE that will be used to identify the design. UNIQUE_ID becomes available only when AUTO has been specified as USERCODE format. If AUTO is not used, this preference has no effect.

Device Support ECP5

Syntax UNIQUE_ID <16-bit hex value>;

or

UNIQUE_ID HEX <16-bit hex value>;

Examples
```
UNIQUE_ID "FFFF";
UNIQUE_ID_HEX "FFFF";
```

USE DIN CELL

Specifies the given register to be used as an input Flip Flop. The default value is TRUE. TRUE means that registers will be moved into I/Os, and FALSE means that registers will be moved out of I/Os. The software distributes the USE DIN CELL setting to the .nmc file after mapping.

Device Support All except MachXO, MachXO2, MachXO3D, MachXO3L, MachXO3D

Syntax USE DIN [TRUE|FALSE] CELL <*cell_name*>;

where:

<*cell_name*> ::= string

Examples
```
USE DIN CELL "din0";
USE DIN FALSE CELL "din0"
```

USE DOUT CELL

Specifies the given register to be used as an output Flip Flop. TRUE means that registers will be moved into I/Os, and FALSE means that registers will be moved out of I/Os. The software distributes the USE DOUT CELL setting to the .nmc file after mapping.

Device Support All except MachXO, MachXO2, MachXO3D, MachXO3L, MachXO3D

Syntax USE DOUT [TRUE|FALSE] CELL <*cell_name*>;

where:

<*cell_name*> ::= string

```
USE DOUT CELL "dout1";
USE DOUT FALSE CELL "dout0"
```

USE EDGE

Runs a clock signal through edge clock resources and minimizes the delay from the clock sources to I/O registers. Use an edge clock resource to route the specified net. The specified net should be a clock net.

When USE EDGE is specified, the Place & Route process will try to use edge clock routing resources, provided that the following conditions are true: the source of the clock is capable of driving an edge clock; all destinations of the clock are capable of receiving an edge clock; and edge clock resources are available in the bank structure.

Device Support LatticeECP2/M

Syntax USE EDGE NET <*net_name*>;

Example The following preference specifies that an edge clock resource should be used when routing a clock net named clk_fast:

```
USE EDGE NET "clk_fast";
```

USE EDGE2EDGE

Enables a clock to route to multiple-edge clocks on the device using a 3-way (left/right/top) bridge. There are two bridges on the left side of device: one for ECLK1 and a second one for ECLK2. Each bridge connects the respective left, right, or top edge clocks.

Device Support LatticeECP3 EA devices, which provide 3-way bridge usage. This preference is not supported for LatticeECP3 E devices.

ECLK1 Bridge For the ECLK1 bridge, the clock can be sourced from a dedicated left PLL PIO, a dedicated path from the left GDLL CLKOP, a dedicated path from the left GPLL CLKOP, a dedicated path from a GDLL slave delay block, or a primary clock input (dedicated PIO, PLL, DLL, CLKDIV, or PCS).

ECLK2 Bridge For the ECLK2 bridge, the clock can be sourced from a dedicated left PLL PIO, a dedicated path from the left GDLL CLKOS, a dedicated path from the left GPLL CLKOS, a dedicated path from a GDLL slave delay block, or a primary clock input (dedicated PIO, PLL, DLL, CLKDIV, or PCS).

EDGE2EDGE vs. PRIMARY2EDGE The EDGE2EDGE preference is intended for well-controlled timing skew when the number of ECLK I/O loads is more than the number of IOs per side using the 3-way (left/right/top) bridge.

The PRIMARY2EDGE preference is useful for implementing a low-skew clock path to the left and right sides of the device. The primary clock must be sourced from a dedicated primary clock source such as a primary clock input, PLL, DLL, CLKDIV, or PCS.

Syntax USE EDGE2EDGE NET <*net_name*>

where:

<*net_name*> is the name (<*string*>) of a specific edge clock net.

Architecture Support The USE EDGE2EDGE preference is supported for MachXO2 devices. It is also supported for LatticeECP3 EA devices, which provide 3-way bridge usage. This preference is not supported for LatticeECP3 E devices.

Recommendations The following guidelines should be followed in using EDGE2EDGE and PRIMARY2EDGE.

- Do not assign the same clock net to both EDGE2EDGE and PRIMARY2EDGE.
- Spread the clock loads of PRIMARY2EDGE and/or EDGE2EDGE signals to appropriate banks by using the LOCATE preference. For example: Locate I/O loads of EDGE2EDGE clock signals to the top, left and right side I/O banks (bank 0, 1, 2, right porting of bank 3, left portion of bank 6, 7) for LatticeECP3 EA devices.

Example The following example defines the tx_clk signal as an EDGE2EDGE signal. In this example, the source for tx_clk is a non primary clock input (dedicated PIO, dedicated path from GDLL, dedicated path from GPLL, dedicated path from GDLL slave delay block).

```
USE EDGE2EDGE NET "tx_clk" ;
```

The following example defines the tx_clk signal as an EDGE2EDGE signal. In this example, the source for tx_clk is a primary clock input (dedicated PIO, PLL, DLL, CLKDIV, or PCS).

```
USE EDGE2EDGE NET "tx_clk" ;
USE PRIMARY NET "tx_clk";
```

See Also ▶"USE PRIMARY2EDGE" on page 1270

USE PRIMARY

Uses a primary clock resource to route the specified net. The specified net should be a clock net. A USE SECONDARY preference overrides a USE PRIMARY for the same net if it appears later in the preference file.

The USE PRIMARY and USE SECONDARY physical preferences are needed when user intervention is required to route nets using primary and secondary clock resources. PAR uses various criteria for selecting the nets, including the number and type of loads, the availability of clock drivers, and the availability of clock trees.

Device Support All

Syntax USE PRIMARY [PURE | DCS] [NET *<net_name>* [*<quadrant_type>*+]];

where:

<net_name> ::= specific clock net name *<string>*

[(quadrant_type)+] ::= [QUADRANT_TL | QUADRANT_TR | QUADRANT_BL | QUADRANT_BR]

QUADRANT_TL ::= Top left

QUADRANT_TR ::= Top right

QUADRANT_BL ::= Bottom left

QUADRANT_BR ::= Bottom right

Note

Currently, a clock cannot be distributed to a set of quadrants where two of the quadrants are diagonal from each other unless the clock is distributed to all quadrants. This means that any set of three quadrants is illegal, as is a set of two quadrants that are diagonal from each other.

PURE ::= Clock spine will be routed without using DCS routing resource

DCS ::= Clock spine will be routed using DCS routing resource

Note

The DCS keyword uses DCS clock spines for routing only. To implement the actual DCS functionality, you must instantiate the DCS in the netlist.

When PURE or DCS is not specified, the clock spine routing can use DCS or PURE resources.

The PURE and DCS keywords are not supported for LatticeSC/M devices.

Examples The following command specifies that a primary clock resource will be used to route a clock net named **clk_fast**:

```
USE PRIMARY NET "clk_fast";
```

The following command specifies that a primary clock resource will be used to route the clock net **bf_clk**. The clock spine will be routed using the DCS routing resource.

```
USE PRIMARY DCS NET "bf_clk";
```

The following command specifies that a primary clock resource will be used to route the clock net. The clock spine will be routed without using the DCS routing resource, and it will use the Top Left Quadrant.

```
USE PRIMARY PURE NET "bf_clk" QUADRANT_TL;
```

USE PRIMARY2EDGE

Note

If there are multiple assignments for the same net, the last preference in the file will take precedence. This is a general condition for all preferences. See "Preference Conflict Resolution" on page 395.

Enables a primary clock to route to a single-edge clock or multiple-edge clocks on the device.

Device Support LatticeECP2/M, LatticeECP3, LatticeSC/M

Syntax USE PRIMARY2EDGE NET <*net_name*>

For LatticeECP2 and LatticeECP3 devices, use this syntax:

USE PRIMARY2EDGE NET <*net_name*>

USE PRIMARY NET <*net_name*>

where:

<*net_name*> is the name (<*string*>) of a specific edge clock net.

The PRIMARY2EDGE preference enables you to select a primary clock and route this clock to a single-edge clock or to multiple-edge clocks. It is useful for implementing a low-skew clock path to all sides of the device. The primary clock must be sourced from a dedicated primary clock source, such as a primary clock input, PLL, DLL, CLKDIV, or PCS.

Architecture Support For LatticeSC devices:

- Banks 1, 4, and 5 support two inputs for primary clocks to route to edge clocks.
- Banks 6 and 7 support two inputs for primary clocks to route to edge clocks for the entire side of the device.
- Banks 2 and 3 support two inputs for primary clocks to route to edge clocks for the entire side of the device.

For LatticeECP2 and LatticeECP3 devices:

- Single-edge and multiple-edge clocks are restricted to the left edge and the right edge.
- A maximum of two multiple-edge clocks are allowed.
- Only two clock signals can be assigned as PRIMARY2EDGE.

Note

Since a primary clock input is used instead of an edge clock input, the clock-injection delay is larger than that of a standard-edge clock.

Recommendations The following recommendations apply to LatticeSC devices:

- Locate any inputs or outputs using the clock to a specific bank. The Place and Route process will find a solution inside the bank, and the bank location will guide the placer.
- Do not include loads of this clock in a PGROUP, since this might cause a resource conflict.
- The same clock should not be assigned to both PRIMARY2EDGE and EDGE2EDGE, else the software will error out. See USE EDGE2EDGE for additional details.
- Locate I/O loads of PRIMARY2EDGE clock signals to the left and right side I/O banks for LatticeECP2/M and LatticeECP3 devices (bank 2, 3, 6, 7).

Examples The following example defines the tx_clk signal as a PRIMARY2EDGE signal for a LatticeSC device.

```
USE PRIMARY2EDGE NET "tx_clk" ;
```

The next example defines the tx_clk signal as a PRIMARY2EDGE signal for a LatticeECP2 or LatticeECP3 device.

```
USE PRIMARY2EDGE NET "tx_clk" ;
USE PRIMARY NET “tx_clk” ;
```

USE SECONDARY

Uses a secondary clock resource to route the specified net. The specified net should be a net driving CLK/CE/LSR inputs. A USE PRIMARY preference overrides a USE SECONDARY for the same net if it appears later in the preference file.

The USE PRIMARY and USE SECONDARY physical preferences are needed when user intervention is required to route nets using primary and secondary clock resources. PAR uses various criteria for selecting the nets, including the number and type of loads, the availability of clock drivers, and the availability of clock trees.

Device Support All except ECP5

Syntax USE SECONDARY [NET <*net_name*> [<*quadrant_type*>+]];

Note

QUADRANT assignments for the USE SECONDARY preference are supported only for LatticeXP and LatticeECP devices.

For LatticeECP3 devices, a net load driven by a secondary clock can be assigned to a particular clock region. Refer to the device's data sheet for more information about clock regions. See Also "REGION" on page 1244.

USE SECONDARY [NET <*net_name*> [(clock_region_site_name) | REGION <*clock_region_name*>]];

where:

<*net_name*> ::= the specific name of a net to be routed using a secondary spine <*string*> (the secondary spine can drive clocks, CEs, resets, or some nodes in switch boxes)

<*quadrant_type*> ::= [QUADRANT_TL | QUADRANT_TR | QUADRANT_BL | QUADRANT_BR]

QUADRANT_TL ::= Top left

QUADRANT_TR ::= Top right

QUADRANT_BL ::= Bottom left

QUADRANT_BR ::= Bottom right

Note

Currently, a clock cannot be distributed to a set of quadrants where two of the quadrants are diagonal from each other unless the clock is distributed to all quadrants. This means that any set of three quadrants is illegal, as is a set of two quadrants that are diagonal from each other.

<clock_region_site_name> ::= the site of a clock region, identified by its location; for example, CLKREG_R2C2

<clock_region_name> ::= The user-defined alias of a clock REGION and its size. The alias is created using the REGION constraint.

Note

REGION <clock_region_name> is optional and can only be used when a separate clock region has been defined using the REGION <clock_region_name> <region_clock_space> preference. When REGION <clock_region_name> <region_clock_space> is used for secondary routing, it must precede the USE SECONDARY preference that references it in the logical preference file.

Using Wildcards For the USE SECONDARY preference, NET names with wildcard expressions are allowed. See "Using Wildcard Expressions in Preferences" on page 533 for details about wildcard usage and range expansion.

Example – secondary clock net loads assigned to a quadrant The following command specifies that a secondary clock resource will be used to route a clock net named clk_lessfast:

Note

If there are multiple assignments for the same net, the last preference in the file will take precedence. This is a general condition for all preferences. See "Preference Conflict Resolution" on page 395.

```
USE SECONDARY NET "clk_lessfast" QUADRANT_TL;
```

Example – secondary clock net loads assigned to a clock region This command specifies that a secondary clock resource will be used to route a clock net named clk1 and that all loads should be constrained to the clock region CLKREG_R5C2.

Note

It is possible to specify both placement of a component group (LOCATE UGROUP) and a net group (USE SECONDARY NET) to the same geographic area of the device. Since a net will drive certain logic, it is possible to make the component and net placements incompatible to each other. If a conflict occurs, the net placement has precedence and component placement will be ignored. Conflicts can be avoided by not using both types of placement constraints. However, there may be situations where net placement alone is not enough control where the logic should be placed

```
USE SECONDARY NET "clk1" "CLKREG_R5C2";
```

See Also ▶"Defining a Clock REGION" on page 521

USERCODE

Specifies a code for storing device data such as firmware version number, manufacturer's ID, programming date, programmer make, pattern code, or JEDEC file checksum. USERCODE assists with record maintenance and product flow control. The USERCODE format can be binary, hexadecimal, or ASCII. For LatticeXP/XP2 and MachXO devices, CHECKSUM can be used as USERCODE. For ECP5 devices, the Auto option is available, which enables you to specify a UNIQUE_ID for the design.

Device Support All

Syntax USERCODE <*usercode_format*> <*usercode*>;

USERCODE CHECKSUM;

USERCODE AUTO;

where:

<*usercode_format*> ::= BIN | HEX | ASCII | AUTO

Specifies one of the available formats for codes: binary (BIN), hexadecimal (HEX), text (ASCII), AUTO

<*usercode*> ::= string

Observe the following guidelines when specifying strings for user codes:

- For binary or BIN format, specify a 32-bit user code string using only 1 or 0 digits.
- For hexadecimal or HEX format, specify an eight-character user code string using only 0 through F values. If you enter a lowercase letter value, it will be converted to uppercase.
- For text or ASCII format, specify a four character user code string using only alpha and numeric values. Characters are case sensitive.
- For LatticeXP/XP2 and MachXO devices, CHECKSUM can be specified as user code in the logical preference file or in the Global Preferences sheet of Spreadsheet View.

Note

If you specify fewer characters than expected, the software will pad the code with zeroes to the correct number of characters. The software will not accept codes that exceed the expected number of characters for a given format.

In addition to the physical preference file (.prf), USERCODE is recorded in the data file: the JEDEC file (.jed) for non-volatile devices or the bitstream file (.bit) for volatile devices. The USERCODE can be read back by means of the JTAG port using the Diamond Programmer, a third-party programmer, or an embedded system.

Examples The following are examples of USERCODE preference usages for the various formats.

```
USERCODE BIN "01010011011001010110110101101001";
USERCODE HEX "53656D69";
USERCODE ASCII "Semi";
USERCODE CHECKSUM;
USERCODE AUTO;
```

Each of these USERCODE preference examples is reflected in the data file. For binary format (BIN), the JEDEC file will show U00000000000000000000010000011111* [bit order[31:0]. Notice that the USERCODE is padded by zeros on the left. For HEX format, the JEDEC file

will contain the code UHEF456ADFF*. For ASCII format, the JEDEC file will contain the code UAanytext*. CHECKSUM will appear as HEX format in the JEDEC file.

For bitstream files, the USERCODE will only be stored in binary, since a bitstream is a binary file.

Refer to the Programmer online Help for more information about the use of USERCODE, CHECKSUM, and data files.

See Also ▶"USERCODE" on page 1337 attribute

VCC_DERATE

Defines the core voltage for derating.

Device Support All

Syntax VCC_DERATE <*volt_spec_vcc*>;

where:

<*volt_spec_vcc*> ::= NOMINAL | PERCENT (+<*number*> | -<*number*>)

Note

For parts that include a regulator, the correct value to enter is the core voltage after the regulator and not the VCC pin voltage for upper voltage parts.

NOMINAL: Nominal VCC core voltage value as specified in Supported Temperature and Voltage Range for Derating topic.

PERCENT (+<*number*> | -<*number*>): Nominal VCC value plus <*number*>% of VCC value or nominal VCC value minus <*number*>% of VCC value. The legal range is +5 to –5 in increments of 1.

Note

Allow up to +10 percent for MachXO2 HC devices.

Example The following preferences include VCC_DERATE that set both temperature and voltage derating for LatticeSC devices.

See Also ▶"Specifying Voltage and Temperature" on page 484

```
TEMPERATURE 25C;
VCC_DERATE NOMINAL;
VCCAUX_DERATE NOMINAL;
VCCIO_DERATE BANK 0 NOMINAL;
VCCIO_DERATE BANK 1 NOMINAL;
VCCIO_DERATE BANK 2 NOMINAL;
VCCIO_DERATE BANK 3 NOMINAL;
VCCIO_DERATE BANK 4 NOMINAL;
VCCIO_DERATE BANK 5 NOMINAL;
VCCIO_DERATE BANK 6 NOMINAL;
VCCIO_DERATE BANK 7 NOMINAL;
```

VCC_NOMINAL

Designates a nominal power supply voltage to a common collector terminal for a given circuit design. It is used to differentiate between chips running at different nominal voltages. The VCC_NOMINAL preference can be used with the VCC_DERATE value to derive a new voltage value. The priority of a given voltage preference would be as follows:

1. VCC_NOMINAL with VCC_DERATE preferences to specify a particular voltage
2. VOLTAGE preference only to specify a particular voltage
3. VCC_NOMINAL preference only to define a standard voltage without derating

Device Support All

Syntax VCC_NOMINAL <*volt_spec_vcc*>;

where:

<*volt_spec_vcc*> ::= (+<*number*>) V

Example The following examples show what occurs in different cases.

In the following case, the VCC_NOMINAL preference is ignored and the software honors the VOLTAGE preference.

```
VCC_NOMINAL 1.2V;
VOLTAGE 1.0V;
```

In the following case, the VOLTAGE preference is ignored because VCC_NOMINAL and VCC_DERATE coexist.

```
VCC_NOMINAL 1.2V;
VOLTAGE 1.0V;
VCC_DERATE PERCENT 5;
```

See Also ▶"Specifying Voltage and Temperature" on page 484

VCC1P2_DERATE

Defines VCC12 core voltage for LatticeSC devices for derating.

Device Support LatticeSC/M

Syntax VCC1P2_DERATE <*volt_spec_vcc1p2*>;

where:

<*volt_spec_vcc1p2*> ::= NOMINAL | PERCENT (+<*number*> | -<*number*>)

NOMINAL: Nominal VCC12 core voltage value as specified in Supported Temperature and Voltage Range for Derating topic.

PERCENT (+<*number*> | -<*number*>): Nominal VCC12 value plus <*number*>% of VCC12 value or nominal VCC12 value minus <*number*>% of VCC12 value. The legal range is +5 to –5 in increments of 1.

Examples The following preferences include VCC12P_DERATE that set both temperature and voltage derating for LatticeSC devices.

```
TEMPERATURE 25C;
VCC_DERATE NOMINAL;
VCC1P2_DERATE NOMINAL;
VCCAUX_DERATE NOMINAL;
VCCIO_DERATE BANK 0 NOMINAL;
VCCIO_DERATE BANK 1 NOMINAL;
VCCIO_DERATE BANK 2 NOMINAL;
VCCIO_DERATE BANK 3 NOMINAL;
VCCIO_DERATE BANK 4 NOMINAL;
VCCIO_DERATE BANK 5 NOMINAL;
VCCIO_DERATE BANK 6 NOMINAL;
VCCIO_DERATE BANK 7 NOMINAL;
```

See Also ▶"Specifying Voltage and Temperature" on page 484

VCCA_DPHY0_DERATE / VCCA_DPHY1_DERATE

VCCA_DPHY0_DERATE / VCCA_DPHY1_DERATE is used to derate the MIPIDPHY power supply. Valid values are from -5% to 5%, starting from a base 1.2V.

Device Support LIFMD/F

Note

Only the UCFBGA64 package type of the LIFMD/F family of devices supports these two constraints.

Syntax VCCA_DPHY0/1_DERATE <volt_spec_vcc>;

where:

<*volt_spec_vcc*> ::= NOMINAL | PERCENT (+<*number*> | -<*number*>)

NOMINAL: 1.2V

PERCENT (+<*number*> | -<*number*>): Nominal VCC value plus <*number*>% of VCC value or nominal VCC value minus <*number*>% of VCC value. The legal range is +5 to -5 in increments of 1.

Example The following preferences include VCCA_DPHY0_DERATE / VCCA_DPHY1_DERATE that sets MIPIDPHY voltage derating for LIFMD/F devices.

```
VCCA_DPHY0_DERATE PERCENT -4;
VCCA_DPHY1_DERATE PERCENT -4;
```

VCCAUX_DERATE

Defines VCCAUX IO voltage for derating.

Device Support All

Syntax VCCAUX_DERATE <*volt_spec_vccaux*>;

where:

<*volt_spec_vccaux*> ::= NOMINAL | PERCENT (+<*number*> | -<*number*>)

NOMINAL: Nominal VCCAUX voltage value as specified in Supported Temperature and Voltage Range for Derating topic.

PERCENT (+<*number*> | -<*number*>): Nominal VCCAUX value plus <*number*>% of VCCAUX value or nominal VCCAUX value minus <*number*>% of VCCAUX value. The legal range is +5 to –5 in increments of 1.

Example The following preferences include VCCAUX_DERATE that set both temperature and voltage derating for LatticeSC devices.

See Also ▶"Specifying Voltage and Temperature" on page 484

VCCIO_DERATE

Defines VCCIO bank voltage for derating.

```
TEMPERATURE 25C;
VCC_DERATE NOMINAL;
VCCAUX_DERATE NOMINAL;
VCCIO_DERATE BANK 0 NOMINAL;
VCCIO_DERATE BANK 1 NOMINAL;
VCCIO_DERATE BANK 2 NOMINAL;
VCCIO_DERATE BANK 3 NOMINAL;
VCCIO_DERATE BANK 4 NOMINAL;
VCCIO_DERATE BANK 5 NOMINAL;
VCCIO_DERATE BANK 6 NOMINAL;
VCCIO_DERATE BANK 7 NOMINAL;
```

Device Support All

Syntax VCCIO_DERATE [BANK <*bank_number*>] <*volt_spec_vccio*>;

where:

The keyword BANK is optional. The <*bank_number*> field specifies the bank number of an I/O bank. The following are the supported I/O bank numbers:

- 0 for Bank0
- 1 for Bank1
- 2 for Bank2
- 3 for Bank3
- 4 for Bank4
- 5 for Bank5
- 6 for Bank6
- 7 for Bank7

<*volt_spec_vccio*> ::= NOMINAL | PERCENT (+<*number*> | -<*number*>)

NOMINAL: Nominal VCCIO value of the buffer type placed in a particular I/O bank. See available data sheets for information on nominal values.

PERCENT (+<*number*> | -<*number*>): Nominal VCCIO value plus or minus the <*number*>% VCCIO value of the buffer type placed in a particular I/O bank. The legal range is +5 to –5 in increments of 1.

Note

Allow up to +10 percent for MachXO2 HC devices.

Examples The following preferences include VCCIO_DERATE that set both temperature and voltage derating of "nominal" for your buffer type.

Like all other preferences, more specific VCCIO_DERATE preferences will override more generalized preferences. For example:

```
TEMPERATURE 25C;
VCC_DERATE NOMINAL;
VCCAUX_DERATE NOMINAL;
VCCIO_DERATE BANK 0 NOMINAL;
VCCIO_DERATE BANK 1 NOMINAL;
VCCIO_DERATE BANK 2 NOMINAL;
VCCIO_DERATE BANK 3 NOMINAL;
VCCIO_DERATE BANK 4 NOMINAL;
VCCIO_DERATE BANK 5 NOMINAL;
VCCIO_DERATE BANK 6 NOMINAL;
VCCIO_DERATE BANK 7 NOMINAL;
```

```
VCCIO_DERATE PERCENT -2;
VCCIO_DERATE BANK 3 PERCENT -5;
```

In this case, the banks 0-2 and 4-7 are covered by -2, while bank 3 is -5. When the preference engine reads those preferences, it will interpret PERCENT -2 for banks 0-2 and 4-7, while it uses PERCENT -5 for bank 3. When the changes are saved to the preference file the preferences will be written as follows:

```
VCCIO_DERATE BANK 0 PERCENT -2;
VCCIO_DERATE BANK 1 PERCENT -2;
VCCIO_DERATE BANK 2 PERCENT -2;
VCCIO_DERATE BANK 3 PERCENT -5;
VCCIO_DERATE BANK 4 PERCENT -2;
VCCIO_DERATE BANK 5 PERCENT -2;
VCCIO_DERATE BANK 6 PERCENT -2;
VCCIO_DERATE BANK 7 PERCENT -2;
```

See Also ▶"Specifying Voltage and Temperature" on page 484

VOLTAGE

Assigns a value for nominal FPGA core voltage. That is the central array and excluding I/Os. If no temperature or voltage preference is specified, the derating factor is 1.0.

Device Support All

Syntax VOLTAGE <*number*>V;

A space between the number and the voltage symbol is optional. See example.

Example The following preference assigns a value for nominal voltage of 1.5 V.

See Also ▶"Specifying Voltage and Temperature" on page 484

```
VOLTAGE 1.5V;
```

HDL Attributes

HDL attributes are constraints that are attached as text to design objects and interpreted by the Diamond software. A design object can be a specific port, component pin, net, instance, instantiation, or even the entire design. An attribute provides information about the object. For example, an attribute might specify where a component in the logical design must be placed in the physical device, or it might specify a frequency constraint for a net that timing-driven place and route will attempt to meet.

The HDL attributes described in this section are those most commonly used as constraints in HDL source files or schematics. They generally apply to all designs and are not specific to any architecture. Unlike preferences that appear in the logical preference file (.lpf), these attributes must appear in the netlist. It is quite common, however, to use both preferences and HDL attributes. When both are included, the Diamond software uses the union of logical preferences and HDL attributes during an FPGA implementation.

HDL attributes are typically declared using comment notation in Verilog HDL or the VHDL Attribute keyword. They can also be assigned in the Schematic Editor using the property feature. HDL attributes are passed to the EDIF netlist through synthesis tools or by editing the netlist directly. Refer to the individual synthesis tool documentation for instructions. When EDIF2NGD reads the EDIF file, the attributes become part of the FPGA native generic database (.ngd).

Note

A comprehensive guide to library element HDL attributes is available in the "FPGA Libraries Reference Guide" on page 1665. For specific applications, refer to the technical notes that are available on the Lattice web site. Editing these library element attributes is not recommended, because they are usually generated from an array of choices that are made for module generation using IPexpress.

See Also ▶"Adding FPGA Attributes to HDL" on page 569

ALU_LOCK_CNT

Convention ALU_LOCK_CNT= 3, 4, 5,...15

Description Lock Count Cycles. Attached to a DLL element. Takes the integer set of 3 to 15.

Device Support LatticeECP3, LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

ALU_UNLOCK_CNT

Convention ALU_UNLOCK_CNT= 3, 4, 5,...15

Description Unlock Count Cycles. Attached to a DLL element. Takes the integer set of 3 to 15.

Device Support LatticeECP3, LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

BBOX

Convention BBOX= x,x

Description Indicates the bounding box or the area given in number of rows and columns for a given UGROUP. The attribute must appear on the same block as the UGROUP attribute. This has replaced the old PBBOX attribute which is also still valid for backwards compatibility.

Device Support All

Syntax BBOX= H,W

where H is height, W is width.

For example, the following parameter indicates a BBOX that is three rows high and seven columns wide.

Figure 37:

```
BBOX= 3,7
```

The examples below show the BBOX attribute with an HGROUP definition. You will also see BBOX with the UGROUP definition.

VHDL Syntax attribute HGROUP: string;
attribute BBOX: string;

VHDL Example Code

```
attribute HGROUP of struct: architecture is
"reg_group";
attribute BBOX  of struct: architecture is "5,5";
```

Verilog Syntax – Precision //pragma attribute x HGROUP <hgroup_name>
//pragma attribute x BBOX <h,w>

Verilog Example Code – Precision

```
//pragma attribute x HGROUP
reg_group
```

```
//pragma attribute x BBOX 5,5
```

Verilog Syntax – Synplify /* synthesis HGROUP= "<hgroup_name>" BBOX= "<h,w>" */;

Verilog Example Code – Synplify `/* synthesis HGROUP= "reg_group" BBOX= "5,5" */;`

More Examples This is used in conjunction with a UGROUP definition, for example in the EDIF you would see BBOX appear as shown the following:

Figure 38:

```
(property UGROUP (string "SAND") )
(property BBOX (string "3, 20") )
```

After mapping, this would result as the following preference in the PRF file:

Figure 39:

```
PGROUP  "SAND"  BBOX  3  20
```

See Also ▶"Adding FPGA Attributes to HDL" on page 569

BLOCKNET

Convention BLOCKNET= name

Description Applied to a net, blocks timing analysis on the net when PAR or TRACE performs a timing analysis on the design. Any paths through the specified net are also blocked from the timing analysis.

When the design is mapped and the MAP program creates a preference file, the BLOCKNET attribute causes MAP to write a BLOCK NET preference into the preference file.

A net on which you attach a BLOCKNET attribute may not exist after the design is mapped, since it may be absorbed into a device logic cell. To avoid losing the net, attach a NOMERGE attribute to the net in addition to the BLOCKNET attribute. If the net is an output of sequential logic, you do not have to attach the NOMERGE attribute; these nets are never absorbed into logic cells.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

BUS

Convention BUS= name

Description This attribute is placed on a net or group of nets and defines a bus structure to be used in the following preferences: BLOCK, MAXDELAY, MAXSKEW, LOCK, PRIORITIZE, USE LONGLINE, and USE HALFLINE. All nets with the same BUS= attributes are identified as belonging to that bus. Translated by the mapper into the DEFINE BUS preference.

Name refers to the name applied to the bus. If attached to multiple nets, the resultant preference would be:

Figure 40:

```
DEFINE BUS <bus_name> NET <net1> NET <net2> NET <netN>;
```

Specifying these attributes in a netlist relieves designers of the task of searching for and identifying components and nets to construct preferences.

Device Support All

See Also ▶"Adding FPGA Attributes to HDL" on page 569

CLAMP

Convention CLAMP= OFF | ON | PCI

Default value of CLAMP for OUTPUT = OFF.

Default value of CLAMP for INPUT = ON if vccio is same as I/O standard.

Default value of CLAMP for INPUT = OFF if vccio is some other value than I/O standard.

Description The CLAMP option can be enabled for each IO independently. The available settings on the bottom edge of the 1200, 2k, 4k, 7k, and 10k devices is PCI or OFF. All other I/O can use the ON or OFF settings only. The CLAMP attribute is not supported for LVCMOS I/O types used in an under-drive or over-drive mode.

Device Support ECP5, MachXO2, MachXO3D, MachXO3L

VHDL Syntax – Synplify ATTRIBUTE CLAMP: string;
ATTRIBUTE CLAMP OF [pin_name]: SIGNAL IS "[mode]";

VHDL Example Code ATTRIBUTE CLAMP OF md: SIGNAL IS "PCI";

Verilog Syntax – Synplify PinType [pin_name] /* synthesis CLAMP="[mode]"*/;
[mode] = OFF, ON, PCI

Verilog Example Code output [4:0] portA /* synthesis CLAMP="ON"*/;

CLKDELAY

Convention CLKDELAY= DEL0|1|2|3

Description A CLKDELAY attribute can be placed on I/O platform registers for specifying cycle stealing. Map will determine the proper delay, ECLKDEL, CLKIDEL, or CLKODEL, based on clock usage. Attributes are attached to the registers themselves.

Related Attributes:

- CLKIDEL
- CLKODEL
- ECLKDEL

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC

VHDL Syntax ATTRIBUTE CLKDELAY : string;
ATTRIBUTE CLKDELAY OF [pin_type]: SIGNAL IS "[signal_name]";

VHDL Example Code
```
attribute clkdelay : string;
attribute clkdelay of input_vector : signal is "DEL0";
```

Verilog Syntax – Precision //pragma attribute [pin_type] CLKDELAY [signal_name]

Verilog Example Code – Precision
```
//pragma attribute load clkdelay del1
```

Verilog Syntax – Synplify PinType [pin_type] /* synthesis CLKDELAY="[signal_name]"*/;

Verilog Example Code – Synplify
```
//input load /* synthesis CLKDELAY="DEL1" */;
```

CLKDELAY Written Out to Preference (.prf) File Here is an example of how CLKDELAY attributes might appear after they are written out to the preference file as the CLKDELAY logical preference in the post-map phase:

```
CLKDELAY CELL "clkdelay0" DEL0 ;
```

See Also CLKDELAY (Logical preference)

- “Adding FPGA Attributes to HDL” on page 569

CLKFB_FDEL

Convention CLKFB_FDEL= 0, 100, 200,...700

Description CLKFB Fine Delay Setting. Attached to a PLL element (e.g., EHXPLLA). Attribute can take the integer value set of 0, 100, 200,...700.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

CLKIDEL

Convention CLKIDEL= DEL0|1|2|3

Description A CLKIDEL attribute may be placed on input DDR registers for specifying cycle stealing and is used in conjunction with ECLKDEL attribute which is used for the EC Clock. The attributes belong on DDR instances.

Related Attributes:

- CLKODEL
- ECLKDEL
- CLKDELAY

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC

VHDL Syntax ATTRIBUTE CLKIDEL : string;
ATTRIBUTE CLKIDEL OF [pin_type]: SIGNAL IS "[signal_name]";

VHDL Example Code
```
attribute clkidel : string;
attribute clkidel of input_vector : signal is "DEL0";
```

Verilog Syntax – Precision //pragma attribute [pin_type] CLKIDEL [signal_name]

Verilog Example Code – Precision)
```
//pragma attribute load clkidel del1
```

Verilog Syntax – Synplify PinType [pin_type] /* synthesis CLKIDEL="[signal_name]"*/;

Verilog Example Code – Synplify)
```
//input load /* synthesis CLKIDEL="DEL2" */;
```

CLKIDEL Written Out to Preference (.prf) File Here is an example of how CLKIDEL attributes might appear after they are written out to the preference file as the CLKDELAY logical preference in the post-map phase:

```
CLKDELAY CELL "clkidel0" DEL0 ;
```

See Also ▶CLKDELAY (Logical preference)

▶ “Adding FPGA Attributes to HDL” on page 569

CLKI_DIV

Convention CLKI_DIV= value

Description Input clock divider setting attached to a PLL element such as EHXPLLB. For LatticeSC/M and LatticeECP2/3 devices, can also be attached to a DLL element such as CIDLLA.

Device Support LatticeSC/M, LatticeECP2/3

Examples For generic examples of how to use FPGA attributes in your HDL, see Adding FPGA Attributes to HDL.

CLKI_FDEL

Convention CLKI_FDEL= 0, 100, 200,...700

Description CLKI Fine Delay Setting. Attached to a PLL element (e.g., EHXPLLA). Attribute can take the integer value set of 0, 100, 200,...700.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

CLKMODE

Convention CLKMODE= mode

Description Valid mode values are SCLK and ECLK. The default is SCLK.

Attached to either a register declaration for an I/O flip-flop, or an I/O flip-flop instance. CLKMODE directs mapper to use a clock by the edge clock routing. When applied with the value ECLK, the register will be clocked by ECLK.

It is recommended that this preference is put on the net closest to the flip-flop.

Device Support LatticeSC/M, LatticeECP2

VHDL Syntax ATTRIBUTE CLKMODE: string;
ATTRIBUTE CLKMODE OF [signal_name]: SIGNAL IS "[value]";

VHDL Example Code
```
attribute CLKMODE : string;
attribute CLKMODE of input1 : signal is "ECLK";
```

Verilog Syntax – Precision //pragma attribute [signal_name] CLKMODE [value]

Verilog Example Code – Precision
```
//pragma attribute input1 CLKMODE ECLK
```

Verilog Syntax – Synplify PinType [signal_name] /* synthesis CLKMODE ="[value]"*/;

Verilog Example Code – Synplify
```
input input1 /* synthesis CLKMODE ="ECLK" */;
```

CLKODEL

Convention CLKODEL= DEL0|1|2|3

Description A CLKODEL attribute may be placed on output DDR registers for specifying cycle stealing and is used in conjunction with ECLKDEL attribute which is used for the EC Clock. The attributes belong on DDR instances.

Related Attributes:

- CLKIDEL
- ECLKDEL
- CLKDELAY

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC

VHDL Syntax ATTRIBUTE CLKODEL : string;
ATTRIBUTE CLKODEL OF [pin_type]: SIGNAL IS "[signal_name]";

VHDL Example Code
```
attribute clkodel : string;
attribute clkodel of input_vector : signal is "DEL0";
```

Verilog Syntax – Precision //pragma attribute [pin_type] CLKODEL [signal_name]

Verilog Example Code – Precision
```
//pragma attribute load clkodel del1
```

Verilog Syntax – Synplify PinType [pin_type] /* synthesis CLKODEL="[signal_name]"*/;

Verilog Example Code – Synplify
```
//input load /* synthesis CLKODEL="DEL2" */;
```

CLKODEL Written Out to Preference (.prf) File Here is an example of how CLKODEL attributes might appear after they are written out to the preference file as the CLKDELAY logical preference in the post-map phase:

```
CLKDELAY CELL "clkodel0" DEL0 ;
```

Figure 41: HDL Example

```
timescale 100ps/10ps
module
clkdiv_test(clk,reset,d_0,d_1,d_2,d_3,d_4,d_5,d_6,d_7,q_0,q_1,q
_2,q_3,q_4,q_5,q_6,q_7);
input clk;
input reset;
input d_0,d_1,d_2,d_3,d_4,d_5,d_6,d_7;

output q_0,q_1,q_2,q_3,q_4,q_5,q_6,q_7;

IFS1P3DX iff0 (.D(d_0), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_0)) /* synthesis CLKMODE = "ECLK" */;
IFS1P3DX iff1 (.D(d_1), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_1)) /* synthesis CLKMODE = "ECLK" */;
IFS1P3DX iff2 (.D(d_2), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_2)) /* synthesis CLKMODE = "ECLK" */;
IFS1P3DX iff3 (.D(d_3), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_3)) /* synthesis CLKMODE = "ECLK" */;
IFS1P3DX iff4 (.D(d_4), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_4)) /* synthesis CLKMODE = "ECLK" */;
IFS1P3DX iff5 (.D(d_5), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_5)) /* synthesis CLKMODE = "ECLK" */;
IFS1P3DX iff6 (.D(d_6), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_6)) /* synthesis CLKMODE = "ECLK" */;
IFS1P3DX iff7 (.D(d_7), .SP(1'b1), .SCLK(clk),
.CD(reset),.Q(q_7)) /* synthesis CLKMODE = "ECLK" */;
endmodule
```

See Also ▶CLKDELAY (Logical preference)

- “Adding FPGA Attributes to HDL” on page 569

CLKOP_DUTY

Convention CLKOP_DUTY= DISABLED | ENABLED

Description CLKOP Duty Cycle Disable/Enable. Attached to a PLL element (e.g., EHXPLLA). Attribute can take DISABLED or ENABLED values.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

CLKOS_DIV

Convention CLKOS_DIV= 1, 2, 3,...63, 64

Description CLKOS Divider Value Select. Attached to a PLL element (e.g., EHXPLLA). Attribute can take the integer values of 1-64.

Device Support LatticeSC/M, LatticeECP2/M, LatticeECP3, ECP5

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

CLKOS_FDEL

Convention CLKOS_FDEL= 0, 100, 200,...700

Description CLKOS Fine Delay Setting. Attached to a PLL element (e.g., EHXPLLA). Attribute can take the integer value set of 0 to 700.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, “Adding FPGA Attributes to HDL” on page 569.

CLKOS_FPHASE

Convention CLKOS_FPHASE= 0, 11.25, 22.5, 45

Description CLKOS Fine Phase Select. Attached to a DLL element (e.g., TRDLLA). The parameters are decimal values 0, 11.25, 22.5, and 45.

Device Support LatticeSC/M, LatticeECP2/M, LatticeECP3, ECP5

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

CLKOS_PHASE

Convention CLKOS_PHASE= 0, 90, 180, 270, 360

Description CLKOS Phase Select. Attached to a PLL element (e.g., EHXPLLA). Attribute can take the integer value set of 0, 90, 180, 270, 360 degrees. CLKOS_PHASE and CLKOS_VCODEL. These two types of delays

are additive to create the final delay. CLKOS_PHASE works in terms of degrees of the VCO period. CLKOS_VCODEL works in terms of full VCO periods.

Device Support LatticeSC/M, LatticeECP2/M, LatticeECP3

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

COARSE

Convention COARSE= CDEL0, CDEL1, CDEL2, CDEL3

Description Course delay. Attached to the DELAY element, this attribute takes CDEL delay parameters. The opposite is FINE.

Device Support LatticeSC/M

See Also ▶"Adding FPGA Attributes to HDL" on page 569

COMP (PLC)

Convention COMP (PLC)= <comp_name>

Description Attached to a block, specifies a name for the device component that is created when the block is mapped. If attached to multiple blocks, indicates that these blocks are to be mapped together in a component with the specified name.

When the device is mapped, the MAP program tries to map all blocks with the same COMP name in a single component, then give the resulting component the name specified by comp_name. If all of the blocks with the attribute cannot be placed in the same component, MAP groups together as many as possible.

If an I/O buffer is assigned a comp_name, the name must match the comp_name on the corresponding pad.

Device Support All

See Also ▶"Adding FPGA Attributes to HDL" on page 569

DCSMODE

Convention DCSMODE= NEG | POS | HIGH_LOW | HIGH_HIGH | LOW_LOW | LOW_HIGH | CLK0 | CLK1

Description Attached to a DCS element to set the particular mode. See the FPGA Libraries Help System for more on usage. The chart below shows possible values.

Figure 42:

Attribute Name	Description	Output		Value	DCS Fuse Settings				
		SEL = 0	SEL = 1		0	1	2	3	4
DCS MODE	Rising edge triggered, latched state is high	CLK0	CLK1	POS	1	0	0	0	0
	Falling edge triggered, latched state is low	CLK0	CLK1	NEG	0	0	0	0	0
	Sel is active high, Disabled output is low	0	CLK1	HIGH_LOW	0	1	0	0	0
	Sel is active high, Disabled output is high	1	CLK1	HIGH_HIGH	1	1	0	0	0
	Sel is active low, Disabled output is low	CLK0	0	LOW_LOW	0	0	1	0	0
	Sel is active low, Disabled output is high	CLK0	1	LOW_HIGH	1	0	1	0	0
	Buffer for CLK0	CLK0	CLK0	CLK0	0	0	1	0	1
	Buffer for CLK1	CLK1	CLK1	CLK1	0	1	0	1	0

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC/ECP2

See Also ▶TN1103 - LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide

- ▶ TN1049 - LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide
- ▶ TN1098 - LatticeSC sysCLOCK and PLL/DLL User's Guide
- ▶ “Adding FPGA Attributes to HDL” on page 569

DELAYTYPE

Convention DELAYTYPE= CFGBIT | DLL | PCLK | ECLK | ECLKINJ

Description This attributes specifies that delay type on the DELAY element.

Device Support LatticeSC/M

See Also ▶“Adding FPGA Attributes to HDL” on page 569

DIFFCURRENT

Convention DIFFCURRENT= NA, 2, 3P5, 4, 6

Description The differential current-source type driver has programmable output current in each mode of operation. The differential current-source type driver is available on the left and right sides only, primary pairs.

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes. This is an attribute that is used with in conjunction IOBUF preference. To see how global preferences are honored for DIFFCURRENT in an IOBUF preference please see that topic.

Device Support LatticeSC/M, LatticeECP/EC

See Also ▶IO_TYPE

- "Adding FPGA Attributes to HDL" on page 569

DIFFDRIVE

Convention DIFFDRIVE= 1.6, 1.65, 1.7, 1.75, 1.81, 1.87, 1.93, 2.0

Description This attribute sets the differential current drive strength for the MINILVDS output standard. An IO bank can only have differential outputs with the same DIFFDRIVE setting. The default value is set to 1.6mA.

Device Support LatticeECP3, ECP5, MachXO2, MachXO3D, MachXO3L

VHDL Syntax ATTRIBUTE DIFFDRIVE : string;

ATTRIBUTE DIFFDRIVE OF [pin_name]: SIGNAL IS "[diffdrive_strength]";

VHDL Example Code `ATTRIBUTE DIFFDRIVE OF portD: SIGNAL IS "1.6";`

Verilog Syntax – Precision //pragma attribute [pin_name] DIFFDRIVE [diffdrive _strength]

Verilog Example Code – Precision `//pragma attribute q_minilvds DIFFDRIVE 1.6`

Verilog Syntax – Synplify PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DIFFDRIVE="[diffdrive_strength]"*/;

Verilog Example Code – Synplify `output mypin /* synthesis IO_TYPE="MINILVDS" DIFFDRIVE="1.6"*/;`

See Also ▶"MULTDRIVE" on page 1314

- "IOBUF Attributes" on page 478

DIFFRESISTOR

Convention DIFFRESISTOR= OFF (default), 120, 150, 220, 420 (Only for differential buffers)

Description Attribute attached to input and output buffers (e.g., IB, OB) and is used to provide differential termination. It is only available for differential I/O types (see IO_TYPE). In case of a differential output termination, this is only available on the primary pads of the PIO. You have the option to turn on the common mode differential termination. See the VCMT attribute for details.

Device Support LatticeSC/M, LatticeECP3

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes. This is an attribute that is used with in conjunction IOBUF preference. To see how global preferences are honored for DIFFRESISTOR in an IOBUF preference please see that topic.

See Also ▶"Adding FPGA Attributes to HDL" on page 569

DIN

Convention DIN= <signal_name> [FALSE | TRUE] [CELL <cell_name>]

Description This attribute instructs map to pack a DIN/DOUT register into PIO/IOLOGIC sites. A DIN attribute must occur on the register rather than on a buffer as for legacy architectures. See also USE DIN CELL logical preference.

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC, LatticeECP2/M

Rules for using DIN/DOUT

- A DIN attribute must occur on the register rather than on a buffer.
- Mapper attempts to map DIN/DOUT register into PIO/IOLOGIC sites. If unsuccessful, an error message is printed and mapper terminates.
- The registers must conform to PIO/IOLOGIC rules in order for them to be mapped properly. If not, error messages are printed.
- DIN will not work when the input flip-flop is driven by combinational logic.
- DOUT will not work when the output flip-flop drives combinational logic other than OS(E)MUX21.

VHDL Syntax ATTRIBUTE DIN : string;
ATTRIBUTE DIN OF [pin_type]: SIGNAL IS "[signal_name]";

VHDL Example Code

```
attribute din : string;
attribute din of input_pin_abc : signal is "TRUE";
```

Verilog Syntax – Precision //pragma attribute [pin_type] DIN [signal_name]

Verilog Example Code – Precision

```
//pragma attribute
input_pin_abc din TRUE
```

Verilog Syntax – Synplify PinType [pin_type] /* synthesis DIN="[signal_name]"*/;

Verilog Example Code – Synplify `//input input_pin_abc /* synthesis din="TRUE" */;`

DIN/DOUT Logical Preference (.lpf) File

```
USE DIN CELL "r" ;
USE DIN FALSE CELL "r" ;
USE DIN TRUE CELL "r" ;
USE DOUT CELL "q" ;
```

DIN/DOUT Written Out to Physical Preference (.prf) File Here is an example of how DIN/DOUT attributes might appear after they are written out to the physical preference file in the post-map phase:

Figure 43:

```
LOCATE COMP "DD1" SITE "29" ;
USE DIN COMP "DD1" ;
LOCATE COMP "QQ2" SITE "75" ;
USE DOUT COMP "QQ2" ;
LOCATE COMP "BIDIPIN" SITE "127" ;
USE DIN COMP "BIDIPIN" ;
USE DOUT COMP "BIDIPIN" ;
LOCATE COMP "QQ5" SITE "180" ;
USE DOUT COMP "QQ5" ;
```

See Also ▶"Adding FPGA Attributes to HDL" on page 569

DOUT

Convention DOUT= <signal_name> [FALSE | TRUE] [CELL <cell_name>]

Description This attribute instructs map to pack a DIN/DOUT register into PIO/IOLOGIC sites. A DIN attribute must occur on the register rather than on a buffer as for legacy architectures. See also USE DOUT CELL logical preference.

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC, LatticeECP2/M

Rules for using DIN/DOUT

- A DIN attribute must occur on the register rather than on a buffer.
- Mapper attempts to map DIN/DOUT register into PIO/IOLOGIC sites. If unsuccessful, an error message is printed and mapper terminates.
- The registers must conform to PIO/IOLOGIC rules in order for them to be mapped properly. If not, error messages are printed.
- DIN will not work when the input flip-flop is driven by combinational logic.

- DOUT will not work when the output flip-flop drives combinational logic other than OS(E)MUX21.

VHDL Syntax ATTRIBUTE DOUT : string;
ATTRIBUTE DOUT OF [pin_type]: SIGNAL IS "[signal_name]";

VHDL Example Code
```
attribute dout : string;
attribute dout of output_pin_abc : signal is "TRUE";
```

Verilog Syntax – Precision //pragma attribute [pin_type] DOUT [signal_name]

Verilog Example Code – Precision
```
//pragma attribute
output_pin_abc dout TRUE
```

Verilog Syntax – Synplify PinType [pin_type] /* synthesis DOUT="[signal_name]"*/;

Verilog Example Code – Synplify
```
//output output_pin_abc /*
synthesis dout="TRUE" */;
```

DIN/DOUT Logical Preference (.lpf) File
```
USE DIN CELL "r" ;
USE DIN FALSE CELL "r" ;
USE DIN TRUE CELL "r" ;
USE DOUT CELL "q" ;
```

DIN/DOUT Written Out to Physical Preference (.prf) File Here is an example of how DIN/DOUT attributes might appear after they are written out to the physical preference file in the post-map phase:

Figure 44:

```
LOCATE COMP "DD1" SITE "29" ;
USE DIN COMP "DD1" ;
LOCATE COMP "QQ2" SITE "75" ;
USE DOUT COMP "QQ2" ;
LOCATE COMP "BIDIPIN" SITE "127" ;
USE DIN COMP "BIDIPIN" ;
USE DOUT COMP "BIDIPIN" ;
LOCATE COMP "QQ5" SITE "180" ;
USE DOUT COMP "QQ5" ;
```

See Also ▶"Adding FPGA Attributes to HDL" on page 569

DRIVE

Convention DRIVE= NA, 2, 4, 8 (default), 12, 16, 24

Description Attached to bidirectional and output buffers (e.g., BB, BBPD, OB, OBW), the drive strength attribute is available for output standards that support programmable drive strength. Refer to the SysIO Usage Guides to see the IOTYPES that support valid drive strength standards.

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes. This is an attribute that is used with in conjunction IOBUF preference. Refer to IOBUF to see how global preferences are honored for DRIVE in an IOBUF preference.

The programmable drive available on a pad will depend on the VCCIO. Also, not all drive strengths are available on an alternate PIO pad. Both the single-ended driver and differential current source driver have programmable drive strength. Only three drive settings are available when operating at 3.3 volts.

Table 140: Valid Drive Strength Table

Drive Strength (mA)	VCCIO 1.2V	VCCIO 1.5V	VCCIO 1.8V	VCCIO 2.5V	VCCIO 3.3V
2	X				
4	X	X	X	X	
8	X	X	X	X	X
12	X	X	X	X	
16		X	X	X	X
24					X

Device Support All

VHDL Syntax ATTRIBUTE DRIVE : string;
ATTRIBUTE DRIVE OF [pin_name]: SIGNAL IS "[drive_strength]";

VHDL Example Code `ATTRIBUTE DRIVE OF portD: SIGNAL IS "8";`

Verilog Syntax – Precision //pragma attribute [pin_name] DRIVE [drive_strength]
[drive_strength] = 2, 4, 8, 12, 16, 20

Verilog Example Code – Precision `//pragma attribute q_lvttl33_20 DRIVE 20`

Verilog Syntax – Synplify PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DRIVE="[drive_strength]" PULLMODE="[mode]" SLEWRATE="[value]"*/;
[drive_strength] = 2, 4, 8, 12, 16, 20

Note: Example given for Pin I/O Type configuration.

Verilog Example Code – Synplify `//output mypin /* synthesis IO_TYPE="LVTTL33D" DRIVE="16" PULLMODE="UP" SLEWRATE="FAST"*/;`

More Examples The following are examples I/O Type settings using the LatticeEC device family for purposes of demonstration. These all include DRIVE attribute examples:

- VHDL source attribute example

  Path: *<install_dir>/examples/fpga/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/VHDL/iovhdl.vhd*

- Verilog source attributes example (Precision)

  Path: *<install_dir>/examples/fpga/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/verilog_mentor/vlogio.v*

- Verilog source attribute example (Synplify)

  Path: *<install_dir>/examples/fpga/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/verilog_synplify/vlogio.v*

Note

There are also example files for other FPGA architectures available in corresponding directories in the <install_dir>/ispcpld/examples path. For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶IO_TYPE

ECLKDEL

Convention ECLKDEL= DEL0|1|2|3

Description Used for the EC Clock, an ECLKDEL attribute may be placed on input DDR registers for specifying cycle stealing. The attribute belongs on DDR instances.

Related Attributes:

- CLKIDEL
- CLKODEL
- CLKDELAY

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC

VHDL Syntax ATTRIBUTE ECLKDEL : string; ATTRIBUTE ECLKDEL OF [pin_type]: SIGNAL IS "[signal_name]";

VHDL Example Code
```
attribute eclkdel : string;
attribute eclkdel of input_vector : signal is "DEL0";
```

Verilog Syntax – Precision //pragma attribute [pin_type] ECLKDEL [signal_name]

Verilog Example Code – Precision
```
//pragma attribute load eclkdel del1
```

Verilog Syntax – Synplify PinType [pin_type] /* synthesis ECLKDEL="[signal_name]"*/;

Verilog Example Code – Synplify `//input load /* synthesis ECLKDEL="DEL2" */;`

ECLKDEL Written Out to Preference (.prf) File Here is an example of how ECLKDEL attributes might appear after they are written out to the preference file as the CLKDELAY logical preference in the post-map phase:

Figure 45:

```
CLKDELAY CELL "eclkdel0" DEL0 ;
```

See Also ▶CLKDELAY (Logical preference)

- ▶ Adding FPGA Attributes to HDL

EQ_CAL

Convention EQ_CAL=value

Description Equalization filtering, available for single-ended inputs on both true and complementary I/Os and for differential inputs on true I/Os. Equalization is required to compensate for the difficulty of sampling alternating logic transitions with relatively slow slew rate.

Value: 0

Device Support LatticeECP3

VHDL Syntax ATTRIBUTE EQ_CAL : string;
ATTRIBUTE EQ_CAL OF [*pin_name*] : signal is “[*value*]”;

VHDL Example Code

```
attribute eq_cal: string;
attribute eq_cal of input1: signal IS "0";
```

Verilog Syntax – Precision //pragma attribute *pin_name* EQ_CAL [*value*]

Verilog Example Code – Precision `//pragma attribute input1 EQ_CAL 0`

Verilog Syntax – Synplify PinType [*pin_name*] /* synthesis EQ_CAL ="[value]"*/;

Verilog Example Code – Synplify `input input1 /* synthesis EQ_CAL ="0"*/;`

EQ_CAL Logical Preference (.lpf) File EQ_CAL = value

Preference File Example EQ_CAL=0 ;

FDEL

Convention FDEL=value

Description Fine delay adjust. Attached to a PLL element (e.g., EHXPLLB). Attribute can take the integer value set of {-8...8).

Device Support LatticeSC/M, LatticeXP/2, LatticeECP/EC

Examples For generic examples of how to use FPGA attributes in your HDL, see Adding FPGA Attributes to HDL.

FINE

Convention FINE= FDEL0, FDEL1, FDEL2,... FDEL47

Description Fine delay. Attached to the DELAY element, this attribute takes FDEL delay parameters. The opposite is COARSE.

Device Support LatticeSC/M

See Also ▶Adding FPGA Attributes to HDL

FIXEDDELAY

Convention FIXEDDELAY= FALSE | TRUE

Description FIXEDDELAY can be used to achieve zero hold time for the input registers when a direct drive primary clock (no PLL) is employed. In order for zero hold time to be achieved, the fixed delay must delay the data by at least as much as the primary clock injection delay.

(MachXO) The MachXO 1K and 2K devices have a programmable input delay element at all the input pins. This programmable delay can be turned on or off. The attribute can be used when the input is tied to an I/O register. The programmable input delay is used to achieve zero hold time. For MachXO 256 and 640 device fixed delay, the software determines whether to turn the delay fuse on or off depending on the tCO constraints. For MachXO 1200 and 2280 programmable delay, the software sets the delay fuse depending on both placement and tCO constraints.

This attribute takes the following string values as shown below and as illustrated in the syntax above:

Values: TRUE, FALSE

Default: FALSE

Device Support LatticeXP/2, MachXO, LatticeECP/EC, LatticeECP3

Note

The FIXEDDELAY attribute can only be applied to input signals of input registers in VHDL or Verilog files. By default, Synplify no longer infers input registers, so when you use Synplify with a VHDL or Verilog file that contains the FIXEDDELAY attribute, the mapper issues an error message.

For the FIXEDDELAY attribute to work with Synplify, you must also add the SYN_USEIOFF attribute to the VHDL or Verilog file to infer I/O registers.

Error Checking in Map Report FIXEDDELAY can be assigned to input signals of registers that are present in the Programmable I/O (PIO) cells. If FIXEDDELAY is assigned to any registers elsewhere, it will cause Map errors. You can see what is happening by viewing the Map report (.mrp) file. In the Design Summary section, you will see a line similar to the following:

Figure 46:

```
Number of PIO FIXEDDELAY:  7
```

This identifies the number of input registers with a FIXEDDELAY=TRUE attribute value. Secondly, the IO (PIO) Attributes section will show any input register that has this FIXEDDELAY attribute set to TRUE.

See Also ▶Adding FPGA Attributes to HDL

FREQUENCY

Convention FREQUENCY= xxx

Description Clock net attribute used to identify the minimum operating frequency for all sequential output to sequential input pins clocked by the specified net. This is translated by the mapper into the FREQUENCY preference. If the net is optimized away, no preference is generated.

The xxx value is expressed in MHz. The resultant preference would be:

Figure 47:

```
FREQUENCY NET <netname> xxx MHz;
```

Device Support All

See Also ▶Adding FPGA Attributes to HDL

GSR

Convention GSR= DISABLED | ENABLED

Description Use to enable/disable the global set/reset (GSR) for all registered elements from these families. For example, this attribute is applicable to registers, PLLs, and memories such as SP8KA, DP8KA and the like.

Device Support All

Examples For specific usage examples for the GSR attribute, see the "Programmable GSR Verilog HDL Example for a Synchronous RTL Module" section in the "How to Use the Global Set/Reset (GSR) Signal" on page 1174 topic. For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

HGROUP

Convention HGROUP= <identifier>

Description Hierarchical grouping construct. HGROUP is used for grouping components that are to be instantiated multiple times. It indicates a logical partition of a group of components that gets translated into a physical partition for PAR.

Note

In Diamond, use the HGROUP attribute instead of PGROUP. This topic remains viable for earlier versions of the software and for purposes of forward compatibility.

Device Support All

VHDL Syntax attribute HGROUP: string;
attribute BBOX: string;

VHDL Example Code:
```
attribute HGROUP of struct: architecture is "reg_group";
attribute BBOX  of struct: architecture is "5,5";
```

Verilog Syntax – Precision //pragma attribute x HGROUP <hgroup_name>
//pragma attribute x BBOX <h,w>

Verilog Example Code – Precision
```
//pragma attribute x HGROUP reg_group
//pragma attribute x BBOX 5,5
```

Verilog Syntax – Synplify /* synthesis HGROUP= "<hgroup_name>" BBOX= "<h,w>" */;

Verilog Example Code – Synplify `/* synthesis HGROUP= "reg_group" BBOX= "5,5" */;`

See Also ▶HGROUP (preference)

- UGROUP (attribute)
- UGROUP (preference)
- REGION (attribute)
- REGION (preference)
- “Adding FPGA Attributes to HDL” on page 569

HULOC

Convention HULOC = <row#col#>

Description HULOC indicates the physical location of the northwest corner of an HGROUP or UGROUP assignment. This should be attached to same block where HGROUP/UGROUP is specified. This attribute must appear on same block as HGROUP/UGROUP. This has replaced the old PLOC attribute which is also still valid for backwards compatibility.

Device Support All

In the following examples, HULOC is used to anchor a UGROUP definition.

VHDL Syntax attribute UGROUP: string;
attribute UGROUP of <object>: label is "<ugroup_name>";
[attribute HULOC: string;
attribute HULOC of <object>: label is "<site_name>";]
[attribute BBOX: string;
attribute BBOX of <object>: label is "<height>,<width>";

VHDL Example Code
```
attribute UGROUP: reg_group;
attribute UGROUP of x: label is "reg_group";
[attribute HULOC: R10C22D;
attribute HULOC of x: label is "R10C22D";]
[attribute BBOX: 9,7;
attribute BBOX of <object>: label is "9,7";]
```

Verilog Syntax – Precision // pragma attribute <object> UGROUP <ugroup_name>
[// pragma attribute <object> HULOC <site_name>]
[// pragma attribute <object> BBOX <height>,<width>]

Verilog Example Code – Precision
```
//pragma attribute x UGROUP
reg_group
//pragma attribute x HULOC R5C19D
//pragma attribute x BBOX 6,4
```

Verilog Syntax – Synplify /* synthesisUGROUP= "<hgroup_name>"
HULOC= "<site_name>"
BBOX= "<h,w>"
*/;

Verilog Example Code – Synplify
```
/* synthesis UGROUP= "reg_group"
HULOC= "R5C19D"
BBOX= "6,4"
*/;
```

More Examples For more generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶HGROUP (attribute)

- ▶ HGROUP (preference)
- ▶ UGROUP (attribute)
- ▶ UGROUP (preference)

HURLOC

Convention HURLOC = <row#col#>

Description HURLOC indicates the northwest corner of a region for a given HGROUP or UGROUP definition. This attribute must appear on the same block as REGION attribute (old attribute is PREGION). Required if REGION exists. This has replaced the old PRLOC attribute which is also still valid for backwards compatibility.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see the "Adding FPGA Attributes to HDL" on page 569 topic.

See Also ▶REGION (attribute)

- ▶ REGION (preference)

HYSTERESIS

Convention HYSTERESIS=SMALL | LARGE | NA

Description The ratioed input buffers have 2 input hysteresis settings. The HYSTERESIS option can be used to change the amount of hysteresis for the PCI, LVTTL and LVCMOS input and bidirectional I/O standards, except for the LVCMOS12 inputs.

The LVCMOS25R33, LVCMOS18R25, LVCMOS18R33, LVCMOS15R25, and LVCMOS15R33 input types do not support HYSTERESIS so this setting is

always disabled for these types. HYSTERESIS is not supported for LVCMOS I/O types when used in an under-drive or over-drive mode.

The HYSTERESIS option for each of the input pins can be set independently when hysteresis is supported for the IO_TYPE assigned to the input pin.

Device Support ECP5, MachXO2, MachXO3D, MachXO3L

VHDL Syntax – Synplify ATTRIBUTE HYSTERESIS: string; ATTRIBUTE HYSTERESIS OF [signal_name]: SIGNAL IS "[SMALL | LARGE | NA]";

VHDL Example Code ATTRIBUTE HYSTERESIS OF q_lvttl33_17: SIGNAL IS " LARGE ";

Verilog Syntax – Synplify /* synthesis attribute HYSTERESIS [of] signal_name [is] [SMALL | LARGE | NA] */;

Verilog Example Code /* synthesis attribute HYSTERESIS of q_lvttl33_17 is LARGE */;

IMPEDANCE

Convention IMPEDANCE= OFF (default), 25, 33, 50, 100

Description This attribute sets the on-chip programmable output impedance. This programmable option can be set for each of the I/O individually. This attribute sets a series termination. Programmable output impedance is not supported for 3.3 volt drivers. In programmable output impedance mode, the driver legs and passgates are calibrated against an external resistor connected on pin XRES. The default is OFF. See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see the "Adding FPGA Attributes to HDL" on page 569 topic.

See Also ▶"IO_TYPE" on page 1310

IMPEDANCEGND

Convention IMPEDANCEGND= OFF (default), 16.7, 20, 25, 33, 50, 100 (Impedance down)

Description Attached to output and bidirectional buffers, this attribute is OFF by default. This attribute sets the on chip series terminations. This programmable option can be set for each of the I/O individually. This attribute sets a pull down series impedance.

The 25, 20 and 16.7 ohm terminations are only available on the primary pads of the PIO, the rest are available for both primary and alternate pads. The impedance for GND can be set to the values of 16.7, 20, 25, 33, 50 and 100.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶"IO_TYPE" on page 1310

IMPEDANCEVCCIO

Convention IMPEDANCEVCCIO= OFF (default), 16.7, 20, 25, 33, 50, 100 (Impedance up)

Description Attached to output and bidirectional buffers, this attribute is OFF by default. This attribute sets the on chip series terminations. This programmable option can be set for each of the I/O individually. This attribute sets a pullup series impedance.

The 25, 20 and 16.7ohm terminations are only available on the primary pads of the PIO, the rest are available for both primary and alternate pads. The impedance for VCCIO can be set to the values of 16.7, 20, 25, 33, 50 and 100.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶"IO_TYPE" on page 1310

INBUF

Convention INBUF= OFF | ON

Description INBUF is a global setting. All unused input buffers are disabled when INBUF is OFF to save power. When you need to perform boundary scan testing, you must turn ON the INBUF attribute. Turning ON this INBUF attribute will turn on the input buffers.

Defaults are OFF for ECP5, LatticeECP/EC and LatticeECP2/M and ON for LatticeXP/2, MachXO, MachXO2. MachXO3D, and MachXO3L/LF. See the chart below for defaults for all device families.

For LatticeECP2/M devices, powersave is always on by default. There is no powersave mode. Built-in intelligence will turn ON powersave when it sees no inputs are used in the design.

Below is a table listing the defaults of INBUF based on device family.

Table 141: INBUF Defaults for Device Families

Device Family	Default for INBUF
ECP5	OFF
LatticeEC	OFF
LatticeXP/LatticeXP2	ON
LatticeECP/LatticeECP2/M	OFF
MachXO	ON
MachXO2	ON
MachXO3D	ON
MachXO3L/LF	ON

This attribute can also be set in the Global Preferences sheet of Spreadsheet View.

Device Support ECP5, LatticeECP/EC, LatticeXP/2, LatticeECP2/M, MachXO, MachXO3D, MachXO2, and MachXO3L/LF.

VHDL Syntax ATTRIBUTE INBUF : string;
ATTRIBUTE INBUF OF [module_name]: entity IS "[value]";

VHDL Example Code
```
ATTRIBUTE INBUF : string;
ATTRIBUTE INBUF OF behave: entity IS "OFF";
```

Verilog Syntax – Precision //pragma attribute module_name INBUF [value]

Verilog Example Code – Precision
```
//pragma attribute pci_top
INBUF OFF
```

Verilog Syntax – Synplify module [module_name] (ports) /* synthesis INBUF= "[value]"*/;

Verilog Example Code – Synplify
```
module pci_top (portA, portB) /*
synthesis INBUF= "OFF" */;
```

More Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

INIT

Convention INIT= value
INIT=ENABLED | DISABLED (LatticeSC Memory)

Description Initializes the look-up table values for all device families. Provides initialization control for some LatticeSC memories.

INIT is required to specify the look-up table values for LUT elements ORCALUT4, 5, 6, 7, or 8.

When attached to LatticeSC RAM elements PDP16KA, SP16KA, and DP16KA, the INIT attribute is set to ENABLED to allow memory initialization. (Back-end tools will set this value if an initialization file is provided.)

See INITVAL and the FPGA Libraries Guide for DP16KA and other memories for details on memory initialization.

See the ORCALUT4 topic in the FPGA Libraries Help for code examples and more information on INIT attribute usage.

Device Support All

Example – ORCALUT4 LUT Value This VHDL example demonstrates the use of INIT with the ORCALUT4 library element.

Figure 48:

```
 component ORCALUT4  port (
      A,B,C,D : In    std_logic;
            Z : Out   std_logic );
   end component;
   attribute INIT: string;
--                                  FEDCBA9876543210
-- attribute INIT of I2: label is "1000000000000000"; --
Z=A*B*C*D
   attribute INIT of I2: label is "1111111111111110"; --
Z=A+B+C+D
```

For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

INITVAL

Convention INITVAL= value

Description Specifies the initialization value for memory elements such as SPR16X2 and DPR16X2. These elements carry prescribed initialization values. For example, DPR16X2 has an initialization value of 0x0000000000000000.

Refer to the "FPGA Libraries Reference Guide" on page 1665.

for more details on memory elements and the INITVAL attribute.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

IO_TYPE

Convention IO_TYPE= buffer type

Description This is used to set the I/O standard for an I/O (input, output, and bidirectional buffers, e.g., IB, OB, and BB). It can takes multiple values that differ slightly depending on which element you are using. The VCCIO required to set these IO standards are embedded in the attribute names. There is no separate attribute to set the VCCIO requirements.

This attribute is used in conjunction with the IOBUF preference. Refer to IOBUF to see how global preferences are honored for IO_TYPE. For valid buffer types, refer to the specific element in the FPGA Libraries Help. See the sysIO usage guides for legal IO_TYPE for inputs and outputs.

Examples of attribute usage for VHDL and Verilog are included in this topic.

Device Support All

Values LVDS, BLVDS25, MLVDS25, RSDS, LVPECL25, LVPECL33, HSTL15_I, HSTL15_II, HSTL15_III, HSTL15_IV, HSTL15D_I, HSTL15D_II, HSTL18_I, HSTL18_II, HSTL18_III, HSTL18_IV, HSTL18D_I, HSTL18D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL33_I, SSTL_II, SSTL33D_I, SSTL33D_II, GTLPLUS15, GTL12, LVTTL33, LVTTL33D, LVCMOS33, LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, PCI33, PCIX33, PCIX15, AGP1X33, AGP2X33, LVCMOS25D, LVCMOS33D, LVCMOS18D, LVCMOS15D, LVCMOS12D

Default LVCMOS25

sysIO Usage Guides Refer to the following usage guides for information on legal combinations. Contact technical support for more detailed information on standard support.

- TN1262, *ECP5 sysIO Usage Guide*
- TN1056, *LatticeECP/EC sysIO Usage Guide*
- TN1088, *LatticeSC PURESPEED I/O Usage Guide*
- TN1102, *LatticeECP2/M sysIO Usage Guide*

- TN1177, *LatticeECP3 sysIO Usage Guide*
- TN1091, *MachXO sysIO Usage Guide*
- TN1202, *MachXO2 sysIO Usage Guide*
- TN1280, *MachXO3L sysIO Usage Guide*
- TN1305, *CrossLink sysI/O Usage Guide*
- FPGA TN-02065, Implementing High-Speed Interfaces with MachXO3D Device
- PURESPEED I/O Technology web page (for LatticeSC/M)
- WP - *Designing 2Gbps Parallel I/O with the LatticeSC FPGA*

VHDL Syntax ATTRIBUTE IO_TYPE : string;
ATTRIBUTE IO_TYPE OF [pin_name]: SIGNAL IS "[type_name]";

Example Code:
```
ATTRIBUTE IO_TYPE OF portA: SIGNAL IS "PCI33";
ATTRIBUTE IO_TYPE OF portB: SIGNAL IS "LVCMOS33";
ATTRIBUTE IO_TYPE OF portC: SIGNAL IS "SSTL33_II";
ATTRIBUTE IO_TYPE OF portD: SIGNAL IS "LVCMOS25";
```

Verilog Syntax – Precision //pragma attribute [pin_name] IO_TYPE [type_name]
[type_name] = (valid I/O type, e.g., LVCMSO18)

Verilog Example Code – Precision
```
//pragma attribute d_lvds25e IO_TYPE LVDS25E
```

Verilog Syntax – Synplify PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DRIVE="[drive_strength]" PULLMODE="[mode]" SLEWRATE="[value]"*/;
[value] = (valid I/O type, e.g., LVCMSO18)

Note: Example given for Pin I/O Type configuration.

Verilog Example Code – Synplify
```
output [4:0] portA /* synthesis IO_TYPE="LVTTL33" DRIVE="16" PULLMODE="UP" SLEWRATE="FAST"*/;
```

More Examples The following examples demonstrate I/O Type settings using the LatticeEC device family:

- VHDL source attribute example

  Path: *<install_dir>/examples/fpga/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/VHDL/iovhdl.vhd*

- Verilog source attributes example (Precision)

  Path: *<install_dir>/examples/fpga/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/verilog_mentor/vlogio.v*

- Verilog source attribute example (Synplify)

  Path: *<install_dir>/examples/fpga/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/verilog_synplify/vlogio.v*

See Also TN1088 - LatticeSC PURESPEED I/O Usage Guide

▶ “Adding FPGA Attributes to HDL” on page 569

LOAD

Convention LOAD= xx.xxx

Description Attached to an output buffer, modifies loading on outputs from the default 50 PFs for timing analysis. Translated by the mapper into the OUTPUT LOAD preference.

The xx.xxx represents the loading on the output in picofarads. The resultant preference would be:

Figure 49:

```
OUTPUT COMP <name> LOAD xx.xxx PF;
```

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

LOC

Convention LOC= site_name (PIN | POS)

Description Specifies a site location for the component that is created when this block is mapped. If attached to multiple blocks, indicates that these blocks are to be mapped together in the specified site.

When the device is mapped, the Map Design process (**map**) first maps blocks to PLCs without looking at LOC attributes. Then it tries to merge all PLCs with the same LOC attribute into a single PLC, and then it assigns this PLC to the site with the specified site name. If all of the blocks with the LOC attribute cannot be mapped into a single component, MAP groups together as many as possible.

When it encounters a LOC preference, MAP also writes a LOCATE preference for the component into the preference file. The LOCATE preference locks the component to the site. When you run PAR, the component cannot be unplaced, moved, swapped, or deleted.

The LOC attribute can be attached to anything that will end up on an I/O cell, and to clocks and internal flip-flops, but it should not be attached to combinational logic that will end up on a logic cell; doing so could fail to generate a locate preference. The LOC attribute overrides register ordering.

Device Support All

VHDL Syntax ATTRIBUTE LOC : string;
ATTRIBUTE LOC OF [pin_name]: SIGNAL IS "[site_name]";

VHDL Example Code
```
ATTRIBUTE LOC : string;
ATTRIBUTE LOC OF output_vector : SIGNAL IS "H5";
```

Verilog Syntax – Precision //pragma attribute [pin_name] LOC [site_name]

Verilog Example Code – Precision
```
// pragma attribute rst LOC H5
```

Verilog Syntax – Synplify PinType [pin_name] /* synthesis LOC="[site_name]"*/;

Verilog Example Code – Synplify
```
//input rst  /* synthesis LOC="H5" */ ;
```

The following examples show slice locations for packing registers:

VHDL Example Code for Register Packing
```
ATTRIBUTE LOC : string;
ATTRIBUTE LOC OF inreg : SIGNAL IS "R2C9D R2C9D R2C8D R2C8D
R2C7D R2C7D R2C6D R2C6D R2C5D R2C5D R2C4D R2C4D R2C3D R2C3D
R2C2D R2C2D";
```

Verilog Example Code for Register Packing (Precision)
```
// pragma
attribute inreg LOC R2C9D R2C9D R2C8D R2C8D R2C7D R2C7D R2C6D
R2C6D R2C5D R2C5D R2C4D R2C4D R2C3D R2C3D R2C2D R2C2D
```

See Also ▶"Conflicting IOBUF Constraints" on page 398

▶ "Adding FPGA Attributes to HDL" on page 569

LOCK_DELAY

Convention LOCK_DELAY= <integer>

Description This is a PLL lock time attribute used for simulation. Valid integer values are 0 to 100. The default value is 100 ns. This attribute is optional. If you wish to enter other than the default value of 100 ns, it can be done by adding this attribute in the DEFPARAM section of the code generated by IPexpress. You can also set this attribute in Spreadsheet View.

Device Support LatticeEC, LatticeECP/2/3, LatticeSC/M, LatticeXP/2

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

MAXDELAY

Convention MAXDELAY= xxx

Description Net attribute that identifies a maximum total delay for a net, bus, or path in the design. Translated by the mapper into the MAXDELAY preference.

The xxx value is expressed in nanoseconds. The resultant preference would be:

Figure 50:

```
MAXDELAY NET  <netname>  xxx NS;
```

Device Support All

MULTDRIVE

Convention MULTDRIVE= 1X, 2X, 3X, 4X

Description This attributes sets the drive strength for individual output buffers for the MINILVDS output standard. Differential outputs with different MULTDRIVE settings can be placed in the same IO bank. The default value is set to 1X.

Device Support LatticeECP3

VHDL Syntax ATTRIBUTE MULTDRIVE : string;

ATTRIBUTE MULTDRIVE OF [*pin_name*]: SIGNAL IS "[*multdrive_value*]";

VHDL Example Code `ATTRIBUTE MULTDRIVE OF portD: SIGNAL IS "1X";`

Verilog Syntax – Precision //pragma attribute [pin_name] MULTDRIVE [multdrive_value]
[diffdrive _strength] = 1X, 2X, 3X, 4X

Verilog Example Code – Precision `//pragma attribute q_minilvds MULTDRIVE 1X`

Verilog Syntax – Synplify The following syntax is used for Pin I/O Type configuration:

PinType [pin_name] /* synthesis IO_TYPE="[type_name]"

DIFFDRIVE="[diffdrive_strength]" MULTDRIVE="[multdrive_value]"*/;

Verilog Example Code – Synplify `output mypin /* synthesis IO_TYPE="MINILVDS" DIFFDRIVE="1.6" MULTDRIVE = "1X"*/;`

See Also ▶"DIFFDRIVE" on page 1294

NOCLIP

Convention NOCLIP

Description Assigned to a net or an instance, indicates that the net or instance cannot be removed as unused logic.

When the design is mapped, the default MAP setting removes any unused logic from the design. In most cases, you will want to do this, since unused logic takes up device components and routing resources unnecessarily. However, when you want to keep unused logic in the design, place a NOCLIP attribute on selected nets, and the nets on which you place the attribute are protected from removal, along with any valid logic connected to these nets.

For a path to be considered valid, it must have a valid input and a valid output.

Valid path inputs include:

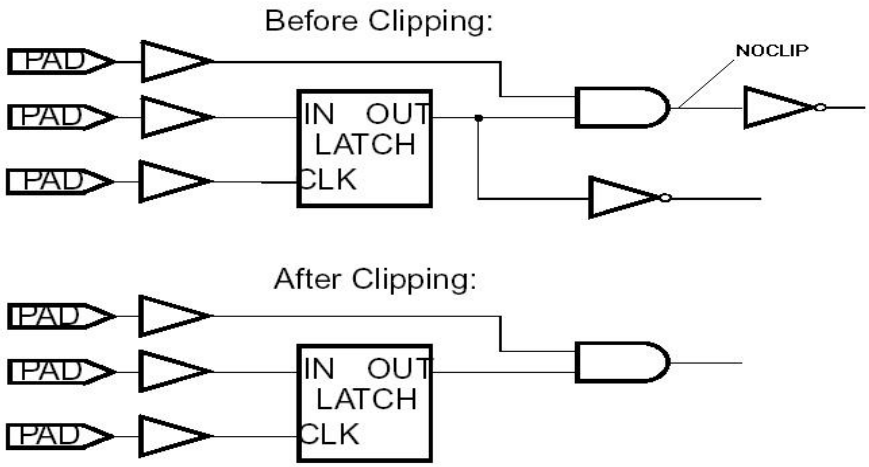
- An input pad or the input portion of a bidirectional pad
- A pullup or pulldown resistor
- An oscillator primitive
- A logic 1 or logic 0 primitive
- A net with a NOCLIP attribute on it

Valid path outputs include:

- An output pad or the output portion of a bidirectional pad
- A net with a NOCLIP attribute on it

If a path is not valid, the MAP program removes its logic as unused. To protect all or part of an invalid path from being removed, add a NOCLIP attribute to the appropriate net. The attribute can act as either a valid input or a valid output, depending on what is necessary to complete the path.

The figure below shows an example of how a NOCLIP attribute affects clipping:



Some notes about using the NOCLIP attribute:

- If you design an on-chip oscillator (by building one from components in the device instead of using an oscillator primitive), the MAP program assumes the logic is unused and removes it because the oscillator logic does not contain a valid driver. To retain your oscillator logic, protect it by placing NOCLIP on the appropriate nets.
- If you design a circuit with a feedback loop, and the looped signal is not driving any other loads but the loop itself, the loop is eliminated as unused logic. If you want to keep the loop in the design, make sure you protect the loop with a NOCLIP attribute.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

NOM_FREQ

Convention NOM_FREQ= value

Description Nominal Frequency selection (MHz) for oscillator elements OSC and OSCD. Values are 2.5 (default), 4.3, 5.4, 6.9, 8.1, 9.2, 10.0, 13.0, 15.0, 20.0, 26.0, 30.0, 34.0, 41.0, 45.0, 55.0, 60.0, and 130.0.

Device Support LatticeECP2/M, LatticeECP3

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

NOMERGE (SAVE)

Convention NOMERGE (SAVE)= <net_name>

Description Assigned to a net, specifies that the net cannot be absorbed into a logic block when the design is mapped. This may happen, for example, if the components connected to each side of a net are mapped into the same logic block. The net may then be absorbed into the block containing the components. When a net is absorbed into a block, it can no longer be seen in the physical design (.ncd) file.

If you want to ensure that a net remains outside of a logic block, attach a NOMERGE attribute to the net. The MAP program then maps the net in a way that ensures you do not “lose” the net inside a logic block.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

NORETIME

Convention NORETIME= string
NORETIME IO

Description An instance with NORETIME attribute will prevent re-timing on its hierarchy. Global NORETIME IO disallows re-timing on I/O registers. I/O register instances named "IFS*" and "OFS*" may be touched by default depending on tSU (INPUT_SETUP) and tCO (CLOCK_TO_OUT) constraints if the attribute "NORETIME IO" is not found.

Device Support All

VHDL Syntax ATTRIBUTE NORETIME : string;
ATTRIBUTE NORETIME OF [instance_name]: LABEL IS "[label_name]";

VHDL Example Code
```
attribute NORETIME  : string;
attribute NORETIME of and2  : label is "IO";
```

Verilog Syntax – Synplify Instance [instance_name] /* synthesis NORETIME="[string]"*/;

Verilog Example Code – Synplify
```
//module and2 (A,B,Z) /*
synthesis NORETIME = 1 */ ;
```

For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

OPENDRAIN

Convention OPENDRAIN= OFF | ON | PLUS

Description You can assign OPENDRAIN to all the LVCMOS and LVTTL standards. OFF is the default.

If OPENDRAIN is used on a bidirectional pin or an output pin, to be valid (does not generate design errors) it either must, 1) be a non-differential output buffer with OPENDRAIN attached, or 2) a non-differential outputZ/bidi buffer in which the "I" (input to output buffer) pin is set to GND and "T" (tristate enable) is live whether the OPENDRAIN attribute is passed or not. The "I" signal can be a GND signal or optimized into output buffer. In cases where a non-differential outputZ/bidi buffer that has the "I" and "T" signals connected together, it does not matter. If the "I" and "T" of outputZ are unique signals and the "I" signal is not set to GND, then OPENDRAIN is not valid.

The OPENDRAIN=PLUS value is available for LatticeECP2, LatticeECP2M, and LatticeXP2 devices. The PLUS value must be used when the user intends to use external pullup greater than VCCIO. When OPENDRAIN = PLUS is used, The software will not use the OPENDRAIN fuse but will instead have to emulate the open drain by tieing the DO to TO of the output buffer. This value of PLUS is only available for OUTPUT and not for BIDI.

Device Support All

VHDL Syntax ATTRIBUTE OPENDRAIN: string;
ATTRIBUTE OPENDRAIN OF [signal_name]: SIGNAL IS "ON";

VHDL Example Code
```
ATTRIBUTE DRIVE OF q_lvttl33_17: SIGNAL IS "ON";
```

Verilog Syntax – Precision //pragma attribute [signal_name] OPENDRAIN [OFF | ON]

Verilog Example Code – Precision
```
//pragma attribute q_lvttl33_17 OPENDRAIN ON
```

Verilog Syntax – Synplify output signal_name /* synthesis OPENDRAIN="ON | OFF" */;

Verilog Example Code – Synplify
```
output q_lvttl33_17 /* synthesis OPENDRAIN="ON" */;
```

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

See Also ▶“IO_TYPE” on page 1310

PCICLAMP

Convention PCICLAMP= OFF | ON

Description Turns on the programmable PCI clamp diode. The PCICLAMP options can be enabled for each I/O independently. For MachXO, The PCICLAMP is only available on the TOP bank on 1K and 2K devices. PCICLAMP are NOT available for the 256 and 640 devices. OFF is the default.

(LatticeXP2) When IO_TYPE is PCI33, then the default for this attribute is ON. You can select PULLMODE to PCICLAMP. If the standard is PCI33 with PCICLAMP=ON, then place in Banks 4, 5. If PCICLAMP=OFF then place in Banks 0, 1, 2, 3, 6, 7, 8. You should use external clamps for PCI outputs placed in Banks 0, 1, 2, 3, 6, 7, 8. This limits the signals to LVCMOS33 and LVTTL33 standards.

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes. This is an attribute that is used in conjunction with the IOBUF preference. To see how global preferences are honored for PCICLAMP in an IOBUF preference, see that topic.

Device Support LatticeECP/EC, LatticeSC/M, LatticeXP/2, MachXO

VHDL Syntax ATTRIBUTE PCICLAMP: string;
ATTRIBUTE PCICLAMP OF [pin_name]: SIGNAL IS "[mode]";

VHDL Example Code
```
ATTRIBUTE PCICLAMP OF md: SIGNAL IS "ON";
```

Verilog Syntax – Precision //pragma attribute [pin_name] PCICLAMP [mode]

[mode] = OFF, ON

Verilog Example Code – Precision
```
//pragma attribute d_lvcmos33_keeper PCICLAMP ON
```

Verilog Syntax – Synplify PinType [pin_name] /* synthesis PCICLAMP="[type_name]"*/;

[mode] = OFF, ON

Verilog Example Code – Synplify
```
output [4:0] portA /* synthesis PCICLAMP="ON"*/;
```

For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

See Also ▶“IO_TYPE” on page 1310

PERIOD

Convention PERIOD = xxx

Description Clock net attribute used to identify a clock period for all pins clocked by the specified net. If the net is optimized away, no preference is generated. Translated by the mapper into the PERIOD preference. The xxx value is expressed in ps (the resultant preference is expressed in ns).

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

PHASEADJ

Convention PHASEADJ= value

Description (LatticeSC/M) VCO Output Phase. Attached to a PLL element (e.g., EHXPLLA), the PHASEADJ attribute adjusts phase relative to CLKOP. PHASEADJ takes the value parameter set of 0, 45, 90, 135, 190, 225, 270, or 315.

(LatticeXP/2, LatticeECP2/M, LatticeECP/EC) Coarse Phase Shift Selection. Attached to a PLL element (e.g., EHXPLLB). Attribute takes the value parameter set of 0, 45, 90, 135, 190, 225, 270, or 315.

Default value is 0.

Device Support LatticeSC/M, LatticeXP/2, LatticeECP2/M, LatticeECP/EC

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

PRIORITY

Convention PR | PRIORITY= 0-100

Description Expressed as an integer, PR and PRIORITY are converted to the PRI attribute with the same value.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

PULLMODE

Convention PULLMODE= UP (default), DOWN, NONE, KEEPER, PCICLAMP

Description Attached to output buffer elements (e.g., OB, OBW) and bidirectional buffers, the PULLMODE attribute mode parameters are UP, DOWN, NONE, KEEPER, PCICLAMP. The PULLMODE options can be enabled for each I/O independently. The KEEPER option is not available for the PULLMODE attribute when VCCIO = 3.3V for LatticeSC/M. The default is UP.

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes. This is an attribute that is used in conjunction with the IOBUF preference. To see how global preferences are honored for PULLMODE in an IOBUF preference please see that topic.

Device Support All

VHDL Syntax ATTRIBUTE PULLMODE: string;
ATTRIBUTE PULLMODE OF [pin_name]: SIGNAL IS "[mode]";

Example Code:
```
ATTRIBUTE PULLMODE OF md: SIGNAL IS "PCICLAMP";
```

Verilog Syntax – Precision //pragma attribute [pin_name] PULLMODE [mode]

[mode] = UP (default), DOWN, NONE, KEEPER, PCICLAMP

Verilog Example Code – Precision
```
//pragma attribute d_lvcmos33_keeper PULLMODE KEEPER
```

Verilog Syntax – Synplify PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DRIVE="[drive_strength]" PULLMODE="[mode]" SLEWRATE="[value]"*/;

[mode] = UP (default), DOWN, NONE, KEEPER, PCICLAMP

Note: Example given for Pin I/O Type configuration.

Verilog Example Code Synplify
```
output [4:0] portA /* synthesis IO_TYPE="LVTTL33_OD" DRIVE="16" PULLMODE="UP" SLEWRATE="FAST"*/ ;
```

More Examples The following are examples I/O Type settings using the LatticeEC device family for purposes of demonstration. These all include PULLMODE attribute examples:

- VHDL source attribute example

  Path: *<install_dir>/examples/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/VHDL/iovhdl.vhd*

- Verilog source attributes example (Precision)

Path: *<install_dir>/examples/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/verilog_mentor/vlogio.v*

- Verilog source attribute example (Synplify)

  Path: *<install_dir>/examples/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/verilog_synplify/vlogio.v*

There are also example files for other FPGA architectures available in corresponding directories in the <install_dir>/ispcpld/examples path.

See Also ▶"IO_TYPE" on page 1310

- "Adding FPGA Attributes to HDL" on page 569

PWRSAVE

Convention PWRSAVE= OFF (default) | ON

Description Powersave feature. This is used on input combinations to reduce the common mode range. When using comparator type inputs pins (pins that use VREF) like HSTL, SSTL etc., when under certain frequencies the user can turn ON the PWRSAVE option to reduce power dissipation on the input pins.

The default for the PWRSAVE feature is OFF so that it will not interfere with JTAG testing of the device. The default is OFF for LVDS and other differential input standards, and default ON for input standards where the full common mode range is not required.

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶"IO_TYPE" on page 1310

RBBOX

Convention RBBOX = H,W (where H is height, W is width)

Description RBBOX indicates the area size a region. This attribute must appear on the same block as REGION attribute (old attribute is PREGION). Required if REGION exists. This has replaced the old PRBBOX attribute which is also still valid for backwards compatibility.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also "HGROUP" on page 1303 (attribute)

- "HGROUP" on page 1220 (preference)
- "UGROUP" on page 1329 (attribute)
- "UGROUP" on page 1263 (preference)
- "REGION" on page 1323 (attribute)
- "REGION" on page 1244 (preference)

REFCIRCUIT

Convention REFCIRCUIT= OFF (default), INTERNAL, EXTERNAL (for differential buffer)

Description (Attached to output buffers (e.g., OB, OBW) and bidirectional buffers, this attribute allows you to choose this internal reference circuit or an external one. The REFCIRCUIT attribute parameters are OFF, INTERNAL, and EXTERNAL. OFF is the default. There is an internal circuit to turn on the bias for LVDS IOTYPE (see IO_TYPE). If you choose the external option then you must provide an external bias.

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes. This is an attribute that is used in conjunction with the IOBUF preference. To see how global preferences are honored for REFCIRCUIT in an IOBUF preference please see that topic.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also "IO_TYPE" on page 1310

REGION

Convention REGION = <identifier>

Description REGION indicates the region to which a given HGROUP or UGROUP belongs. This attribute must appear on a block that has a HGROUP or UGROUP attribute. This has replaced the old PREGION attribute which is also still valid for backwards compatibility.

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see Adding FPGA Attributes to HDL.

See Also ▶"HGROUP" on page 1303 (attribute)

- "HGROUP" on page 1220 (preference)
- "UGROUP" on page 1329 (attribute)
- "UGROUP" on page 1263 (preference)
- "REGION" on page 1244 (preference)

RST_PULSE

Convention RST_PULSE= <integer>

Description This is an attribute used with the Power Up Reset (PUR) element used for setting a required reset pulse length for this signal. You can pass a required numerical value such as 10 ns or 100 ns. The default value for this attribute is 1 ns.

Device Support All

Verilog HDL Example
```
PUR     PUR_INST (.PUR (<powerup reset sig>));
defparam PUR_INST.RST_PULSE = 10;
```

VHDL HDL Example
```
PUR_INST : PUR
generic map (RST_PULSE => 10)
port map (PUR => <signal>);
```

The RST_PULSE parameter in the PUR element instantiations for Verilog and VHDL above shows a a value of 10 ns assigned to set the required reset pulse length.

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

SLEW

Convention SLEW= 1-100

Description Attached to an output buffer or pad, specifies the relative speed (slew rate) of the output driver, with 1 indicating the slowest and 100 the fastest.

The SLEW attribute applies if the I/O cell output slew rate can be programmed in the device to which the design is mapped. If so, MAP programs the output

driver for the specified slew rate in the cell into which the component is mapped. The SLEW attribute is interpreted in this way:

Table 142: Slew Rate Settings

SLEW Attribute Setting	Programmed Slew Rate
0-33	SLEWLIM (Slow)
34-66	SINKLIM (Medium)
67-100	FAST

Device Support All

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

SLEWRATE

Convention SLEWRATE= FAST | SLOW | NA (LatticeSC/M only)

Description Attached to all output buffers (e.g., OB, OBW) and bidirectional buffers, the SLEWRATE attribute mode parameters are FAST or SLOW for all families except LatticeSC/M which also has an NA value. The NA value refers to a "non applicable" condition for some I/O standards in the LatticeSC device family only.

Each I/O pin has an individual slew rate control. This allows designer to specify slew rate control on pin by pin basis. This slew rate control affects both the rising edge and the falling edges. The default condition for SLEWRATE is SLOW for all I/O standards except for LatticeECP3 devices, where the default SLEWRATE is FAST.

See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes. This is an attribute that is used in conjunction with the IOBUF preference. To see how global preferences are honored for SLEWRATE in an IOBUF preference please see that topic.

Device Support All

VHDL Syntax ATTRIBUTE SLEWRATE: string;
ATTRIBUTE SLEWRATE OF [pin_name]: SIGNAL IS "[value]";

VHDL Example Code
```
ATTRIBUTE SLEWRATE OF portA: SIGNAL IS "SLOW";
```

Verilog Syntax – Precision //pragma attribute [pin_name] SLEWRATE [value]

```
[value] = FAST, SLOW
```

Verilog Example Code – Precision `//pragma attribute portA SLEWRATE SLOW`

Verilog Syntax – Synplify PinType [pin_name] /* synthesis IO_TYPE="[type_name]" DRIVE="[drive_strength]" PULLMODE="[mode]" SLEWRATE="[value]"*/;

[value] = FAST, SLOW

Note: Example given for Pin I/O Type configuration.

Verilog Example Code – Synplify `output [4:0] portA /* synthesis IO_TYPE="LVTTL33_OD" DRIVE="16" PULLMODE="UP" SLEWRATE="FAST"*/ ;`

More Examples The following are examples I/O Type settings using the LatticeEC device family for purposes of demonstration. These examples all contain SLEWRATE attributes:

- VHDL source attribute example

  Path: *<install_dir>/examples/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/VHDL/iovhdl.vhd*

- Verilog source attributes example (Precision)

  Path: *<install_dir>/examples/LatticeEC/preferences_attributes/iotype_drive_pullmode_slewrate/verilog_mentor/vlogio.v*

- Verilog source attribute example (Synplify)

  Path: *<install_dir>/examples/LatticeEC/preferences_attributes/*

See Also "Adding FPGA Attributes to HDL" on page 569

SMI_OFFSET

Convention SMI_OFFSET= hexadecimal number

Description System Management Interrupt Offset. Attached to DLL element, the attribute is set with a default offset start value of 12h'410'.

Device Support LatticeECP2/M, LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

SPREAD_DRIFT

Convention SPREAD_DRIFT= 1, 2, 3

Description Spread Spectrum BW. Attached to the PLLA element, the SPREAD_DRIFT attribute takes integer parameters of 1, 2, or 3.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see “Adding FPGA Attributes to HDL” on page 569.

TERMINATEGND

Convention TERMINATEGND= OFF (default), 50, 100, 120 (Termination Down)

Description This attribute determines the on-chip parallel input and output termination. It is available for certain I/O types. Refer to the sysIO Usage Guides to see the I/O types that support this parallel termination to GND. TERMINATEGND is available for each individual pin in the bank, independent of all other pins in the bank.

Attached to output buffer elements—for example, OB, OBW— the TERMINATEGND attribute is OFF by default. You can attach the resistance parameters for 50, 100, and 120 for termination down. See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes.

Device Support LatticeSC/M

Split Termination If you wish to split termination—for example, a parallel termination of type 50 ohm to VCCIO/2—you must turn on both the TERMINATEVCCIO and TERMINATEGND to a value of 100ohm to get the termination you require.

Parallel Termination The following table shows all the parallel terminations supported for LatticeSC/M and the required values for the TERMINATEGND and TERMINATEVCCIO attributes.

Table 143: Parallel Termination

Input & Output Parallel Termination	TERMINATEVCCIO	TERMINATEGND
DEFAULT	OFF	OFF
120 to VCCIO	120	OFF
120 to GND	OFF	120
60 to VCCIO/2	120	120
100 ohm to GND	OFF	100
100 ohm to VCCIO	100	OFF
50 ohm to GND	OFF	50

Table 143: Parallel Termination

Input & Output Parallel Termination	TERMINATEVCCIO	TERMINATEGND
50 ohm to VCCIO	50	OFF
50 ohm to VCCIO/2	100	100
33 ohm to VCCIO	50	OFF
33 ohm to GND	OFF	50
25 ohm to VCCIO/2	50	50

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶"TERMINATEVCCIO" on page 1328

▶ "IO_TYPE" on page 1310

TERMINATEVCCIO

Convention TERMINATEVCCIO= OFF (default), 50, 100, 120 (Termination Up)

Description This attribute determines the on-chip parallel input and output termination. It is a programmable option that is available for certain I/O types. Refer to the sysIO Usage Guides to find the I/O types that support this parallel termination to VCCIO. This attribute is available for each individual pin in the bank. The VCCIO to which the termination is available depends on the VCCIO of the bank.

Attached to output buffer elements (e.g., OB, OBW), the TERMINATEVCCIO attribute is by default OFF. You can attach the resistance parameters for 50, 100, and 120 for termination up. See the Split Termination and Parallel Termination section in the TERMINATEGND topic. See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶"TERMINATEGND" on page 1327

▶ "IO_TYPE" on page 1310

TERMINATEVTT

Convention TERMINATEVTT= OFF (default), 40, 50, 60, 75, 120, 150, 210

Description This attribute sets the on-chip input parallel termination to VTT. This programmable option can be set for each of the I/O individually. When VCCIO is 3.3V, parallel input termination to VTT is still supported in the bank, as long as the PAD itself is not at 3.3 volts. See the LatticeSC PURESPEED I/O Usage Guide for more information on this and related attributes.

Device Support LatticeSC/M, LatticeECP3

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶"IO_TYPE" on page 1310

TERMINATION

Sets the on-chip input parallel termination to VCCIO/2. This parallel termination is achieved using a programmable Thevenin termination scheme of 50/75/150 ohms to VCCIO/2.

Device Support ECP5

Syntax TERMINATION=OFF | ON;

UGROUP

Convention UGROUP = <identifier>

Description Universal grouping construct. Use the UGROUP attribute to group blocks within different hierarchies or with no hierarchy. UGROUP differs from HGROUP attributes in that its identifier is not changed by pre-appending the hierarchy and the block instance. See "UGROUPs in the HDL" on page 394.

You can add UGROUP anchor and bounding box information to the HDL with the HULOC and BBOX attributes, or you can add this information later using the Group sheet in Spreadsheet View. Alternatively, you can allow a UGROUP to float within a REGION.

Device Support All

Syntax UGROUP="<ugroup_name>"
[HULOC="<site_name>"]
[BBOX="<height>,<width>"]

where:

<ugroup_name> is a user-defined name.

<site_name> is a row/column Slice D location of the target device in the format R<n>C<m>D, an EBR site in the format EBR_R<n>C<m>, or a DSP site in the format DSP_R<n>C<m>.

<height> is the number of device rows of a rectangular bounding box.

<width> is the number of device columns of a rectangular bounding box.

Note

For certain MachXO2 devices, the first EBR site on the left is positioned at column 1 and cannot be included in an anchored group or region that contains SLICEs. The column 1 EBR site is not aligned with SLICEs, which begin at column 2. Affected devices include LCMXO2-1200/4000/7000/10000.

The VHDL and Verilog examples show anchored UGROUPs. Completely floating HGROUPs would preclude the use of the HULOC attribute that specifies the anchor point at a specific site location.

VHDL Syntax attribute UGROUP: string;
attribute UGROUP of <object>: label is "<ugroup_name>";
[attribute HULOC: string;
attribute HULOC of <object>: label is "<site_name>";]
[attribute BBOX: string;
attribute BBOX of <object>: label is "<height>,<width>";]

VHDL Example Code
```
attribute UGROUP: reg_group;
attribute UGROUP of x: label is "reg_group";
[attribute HULOC: R10C22D;
attribute HULOC of x: label is "R10C22D";]
[attribute BBOX: 9,7;
attribute BBOX of <object>: label is "9,7";]
```

Verilog Syntax – Precision // pragma attribute <object> UGROUP <ugroup_name>
[// pragma attribute <object> HULOC <site_name>]
[// pragma attribute <object> BBOX <height>,<width>]

Verilog Example Code – Precision
```
//pragma attribute x UGROUP
reg_group
//pragma attribute x HULOC R5C19D
//pragma attribute x BBOX 6,4
```

Verilog Syntax – Synplify /* synthesis UGROUP= "<hgroup_name>"
HULOC= "<site_name>"
BBOX= "<h,w>"
*/;

Verilog Example Code – Synplify
```
/* synthesis UGROUP= "reg_group"
HULOC= "R5C19D"
BBOX= "6,4"
```

```
*/;
```

Note

Synthesis directives to preserve hierarchy are typically needed to retain the instances constrained by the UGROUP attribute.

Example of Floating UGROUP Attribute Given a Verilog module instance:

Figure 51:

```
rotate rotate_1 (q, reg_out, clk, r_l, rst)
  /* synthesis UGROUP= ”rot_group”
  */ ;
  // pragma attribute rotate_1 UGROUP rot_group;
  // pragma attribute rotate_1 hierarchy preserve
```

the resulting .prf file will contain the following in the schematic section after mapping:

Figure 52:

```
PGROUP "rot_group"
  COMP "rotate_1/SLICE_0"
  COMP "rotate_1/SLICE_1"
  COMP "rotate_1/SLICE_2"
  COMP "rotate_1/SLICE_3";
```

Example of Anchored and Bounded UGROUP Attribute Given a Verilog module instance:

Figure 53:

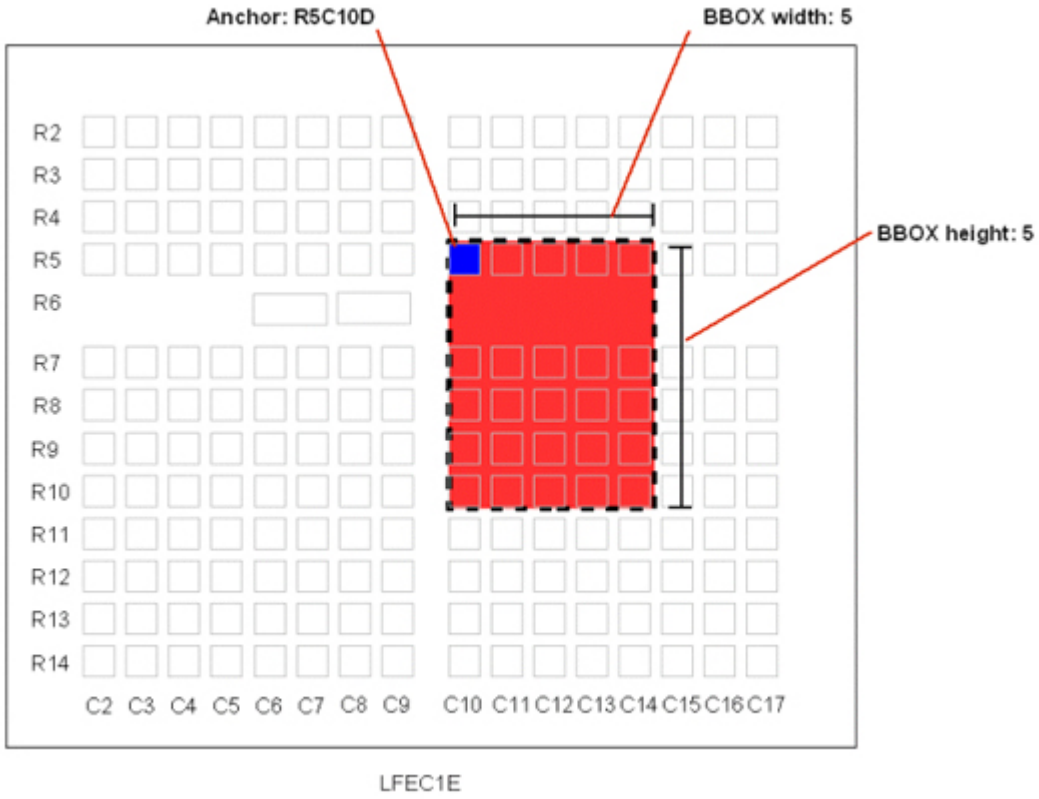
```
rotate rotate_1 (q, reg_out, clk, r_l, rst)
  /* synthesis UGROUP= “rot_group”
     HULOC= “R5C10D”
     BBOX= “5,5”
  */ ;
  // pragma attribute rotate_1 UGROUP     rot_group
  // pragma attribute rotate_1 HULOC      R5C10
  // pragma attribute rotate_1 BBOX       5,5
  // pragma attribute rotate_1 hierarchy  preserve
```

the resulting PRF file will contain the following in the schematic section after mapping:

Figure 54:

```
PGROUP "rot_group" BBOX 5 5
  COMP "rotate_1/SLICE_0"
  COMP "rotate_1/SLICE_1"
  COMP "rotate_1/SLICE_2"
  COMP "rotate_1/SLICE_3";
LOCATE PGROUP "rot_group" SITE "R5C10D" ;
```

The figure below illustrates the rot_group PGROUP in the context of an LFEC1E FPGA device.



Example of Anchored and Bounded UGROUP Attribute on Multiple Instances When the HDL includes multiple instances of the same UGROUP, it is good practice to declare HULOC and BBOX only once to ensure that no conflicting constraints will occur. Given Verilog module instances:

Figure 55:

```
reg pipest_1 (q, d0, clk, r_l, rst)
 /* synthesis UGROUP=   "reg_group"
    HULOC="R5C15D"
    BBOX="5,5"
 */ ;
 // pragma attribute pipest_1 UGROUP reg_group
 // pragma attribute pipest_1 HULOC R5C15D
 // pragma attribute pipest_1 BBOX 5,5
 // pragma attribute pipest_1 hierarchy preserve

 reg pipest_2 (q, reg_out1, clk, r_l, rst)
 /* synthesis                UGROUP=   "reg_group"
 */ ;
 // pragma attribute pipest_2  UGROUP  reg_group
 // pragma attribute pipest_2  hierarchy preserve

 reg pipest_3 (q, reg_out2, clk, r_l, rst)
 /* synthesis                  UGROUP= "reg_group"
 */ ;
 // pragma attribute pipest_3  UGROUP  reg_group
 // pragma attribute pipest_3  hierarchy preserve
```

the resulting PRF file will contain the following in the schematic section after mapping:

Figure 56:

```
PGROUP "reg_group" BBOX 5 5
  COMP "pipest_1/SLICE_4"
  COMP "pipest_1/SLICE_5"
  COMP "pipest_1/SLICE_6"
  COMP "pipest_1/SLICE_7"
  COMP "pipest_2/SLICE_8"
  COMP "pipest_2/SLICE_9"
  COMP "pipest_2/SLICE_10"
  COMP "pipest_2/SLICE_11"
  COMP "pipest_3/SLICE_12"
  COMP "pipest_3/SLICE_13"
  COMP "pipest_3/SLICE_14"
  COMP "pipest_3/SLICE_15";
LOCATE PGROUP "reg_group" SITE "R5C15D" ;
```

Example of UGROUP Attribute with a Register Declaration Given a Verilog module instance:

Figure 57:

```
 reg [7:0] q /* synthesis UGROUP=rot_reg_group */;
```

the resulting PRF file will contain the following in the schematic section after mapping:

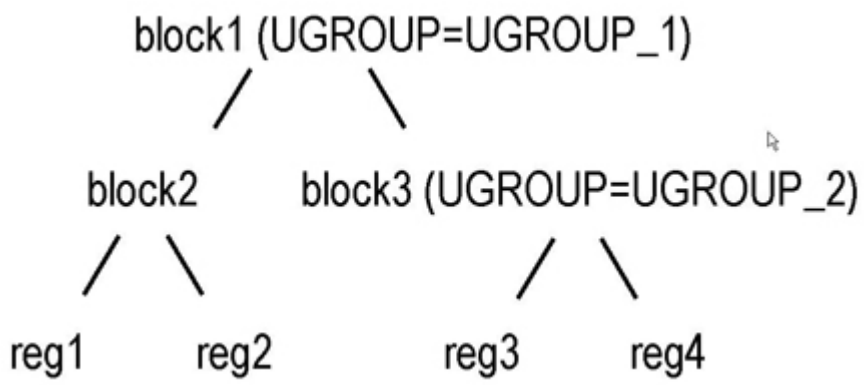
Figure 58:

```
PGROUP "rot_group" BBOX 5 5
  COMP "rotate_1/SLICE_0"
  COMP "rotate_1/SLICE_1"
  COMP "rotate_1/SLICE_2"
  COMP "rotate_1/SLICE_3";
LOCATE PGROUP "rot_group" SITE "R5C15D" ;
```

UGROUP Attribute Usage Rules and Restrictions Observe the following conditions for proper UGROUP attribute usage:

- All elements within the UGROUP'd block belong to that particular UGROUP.
- Nested UGROUP blocks are considered as unique individual UGROUPs.
- All blocks belong to the UGROUP attached to their nearest ancestor in the hierarchy.
- Unlike PGROUP, the mapper will not append the hierarchical path plus the block instance name.
- UGROUP differs from HGROUP in that pre-appending the hierarchy and the block instance does not change its identifier.

For example:



REG1 and REG2 belong to UGROUP_1.
REG3 and REG4 belong to UGROUP_2.

Also, assuming that REG1, REG2 are mapped to PFU_0 and that REG3, REG4 are mapped to PFU_1, the resulting preferences generated in the preference file are as follows:

Figure 59:

```
PGROUP "UGROUP_1"
COMP PFU_0;
PGROUP "UGROUP_2"
COMP PFU_1;
```

If block1 has a UGROUP that is also located, HULOC = R1C1, then the additional preference

Figure 60:

```
LOCATE PGROUP "UGROUP_1" SITE "R1C1"
```

will appear in the preference file.

If block1 has REGION=ONE, HURLOC=R1C1, RBBOX=2,2, then the preferences generated will be as follows:

Figure 61:

```
REGION "ONE" "R1C1" 2 2;
PGROUP "UGROUP_1"
COMP PFU_0;
PGROUP "UGROUP_2"
COMP PFU_1;
LOCATE PGROUP "UGROUP_1" REGION "ONE"
LOCATE PGROUP "UGROUP_2" REGION "ONE"
```

Note: Like HGROUPs, UGROUPs are also resolved into the PGROUP physical preference.

For backward compatibility of existing HDL designs, the property filter file will translate the old attributes to the new set. For HDL attributes, NGDBUILD will resolve HGROUP identifiers on hierarchical blocks, but it will not push the identifier down onto all individual blocks in the hierarchy.

See Also ▶"HGROUP" on page 1303 (attribute)

- "BBOX" on page 1283 (attribute)
- "HGROUP" on page 1220 (preference)
- "UGROUP" on page 1263 (preference)
- "REGION" on page 1323 (attribute)
- "REGION" on page 1244 (preference)

USERCODE

Convention USERCODE=<string>

Description Attached to the top of the design, specifies a binary, hexadecimal, or ASCII usercode for storing device data such as firmware version number, manufacturer's ID, programming date, programmer make, pattern code, or JEDEC file checksum. Usercode definitions at lower hierarchy levels will be ignored. A USERCODE preference in the logical preference file (.lpf) will override the USERCODE attribute.

Device Support All

Guidelines Observe the following guidelines for the usercode format:

- For binary format, specify a string beginning with the characters "0b" followed by 32 binary digits.
- For hexadecimal format, specify a string beginning with the characters "0h" followed by eight hexadecimal values. If you enter a lowercase letter value, it will be converted to uppercase.

- For text or ASCII format, specify a four-character string using only alpha and numeric values. Characters are case sensitive.
- For LatticeXP/XP2 and MachXO devices, you can specify CHECKSUM as USERCODE.

VHDL Syntax attribute usercode : string;

VHDL Example Code
```
architecture behave of prep1 is
attribute usercode : string;
attribute usercode of behave : architecture is
"0b00001111000011111111000011110000";
```

Verilog Syntax – Precision // pragma attribute USERCODE <string>

Verilog Example Code – Precision
```
module count(c,clk,rst);
// pragma attribute count USERCODE "0h0123ABCD"
```

Verilog Syntax – Synplify /* USERCODE=<string> */;

Verilog Example Code – Synplify
```
module count(c,clk,rst)/*
Synthesis USERCODE="0h0123ABCD"*/;
```

See Also ▶"USERCODE" on page 1273 Preference

VCMT

Convention VCMT= OFF (default) | VCMT | VTT | DDR_II

Description Attached to input buffer elements (e.g., IB), assigning value types allows you to enable a common mode voltage, that is, choose between common mode termination or VTT termination. For the DDR memory interface, the attribute allows you to choose DDT_II as an option. DDT_II allows Parallel input termination to VTT when receiving and series output termination during transmittal.

This is available for all the differential and complementary inputs. Refer to the sysIO Usage Guides for more details.

Device Support LatticeSC/M

Examples For generic examples of how to use FPGA attributes in your HDL, see "Adding FPGA Attributes to HDL" on page 569.

See Also ▶"IO_TYPE" on page 1310

Lattice Synthesis Engine (LSE) Constraints

The Lattice Synthesis Engine (LSE) enables you to use constraints that are directly interpreted by the synthesis engine. This new category of constraints includes “Synopsys Design Constraints (SDC)” on page 1339, such as create_clock, and “Lattice Synthesis Engine-Supported HDL Attributes” on page 1349 such as syn_black_box. These constraints influence the optimization or structure of the output netlist.

See Also ▶“Integrated Synthesis” on page 583

- “Applying Lattice Synthesis Engine Constraints” on page 366

Synopsys Design Constraints (SDC)

This section describes the Synopsys® Design Constraint (SDC) language elements for timing-driven synthesis that are supported by the Lattice Synthesis Engine (LSE). When you use LSE, these SDC constraints are saved to a Lattice Design Constraints file (.ldc). A new .ldc file can be created and edited using the LDC Editor or the Source Editor. You can select the .ldc file to be the active synthesis constraint file for an implementation.

When the “Use LPF Created from SDC in Project” option is set to True (-lpf 1) in the active strategy file, the constraints contained in the .ldc file will be output into a synthesis logical preference file:

<*project_name*>_<*implementation_name*>.lpf.

The SDC constraints will drive optimization of the design if LSE’s Optimization Goal is set for timing in the active strategy file.

Considerations about LSE Timing The module delay in LSE is based on the primitive level, before logic is packed into a slice. Therefore, the delay might be different from the slice-based MAP or PAR in the TRACE analysis report (.twr). Also, some elements are modeled as black boxes in LSE; for example, EFB in MachXO2.

The routing delay algorithm in LSE is different from the estimated routing delay in Map or Place.

The current LSE timing does not take the PLL/DLL frequency or phase shift properties into account. It also does not model the different IO_TYPE in the PIO. Therefore, it is necessary to adjust the timing constraint. For example, you can explicitly include a timing constraint on the PLL outputs with the phase-shift property.

See Also ▶“Design Objects” on page 1340

- “create_clock” on page 1341
- “create_generated_clock” on page 1343
- “set_clock_groups” on page 1344
- “set_false_path” on page 1345
- “set_input_delay” on page 1346

- "set_max_delay" on page 1347
- "set_min_delay" on page 1347
- "set_multicycle_path" on page 1348
- "set_output_delay" on page 1348
- "Integrated Synthesis" on page 583
- "Applying Lattice Synthesis Engine Constraints" on page 366

Design Objects

Design objects can be referred to in the SDC as a single object, or as a collection of objects. Single objects must be referred to as a collection of a single object. The current implementation of the SDC commands allows only single objects in the collection. The exception to this are the all_* commands.

In the examples below, <*target name*> is a regular expression that matches just one object only.

Clock Object

SDC Collection	Description
[get_clocks <*target name*>]	<*target name*> is the name of the clock. Clock is either given a name or gets its name from the port/net on which it is defined.
[all clocks]	Collection of all clocks in the design.

Examples: [get_clocks clock_fast]

[all_clocks]

Port Object

SDC Collection	Description
[get_ports <*target name*>]:	Collection of a design port that matches <*target name*>
[all_inputs]	Collection of all design input ports
[all_outputs]	Collection of all design output ports

Examples: [get_ports indata*]

[all_inputs]

Cell Object

SDC Collection	Description
[get_cells <*target name*>]:	Collection of a design instance that matches <*target name*>

Net Object

SDC Collection	Description
[get_nets <*target name*>]:	Collection of a design net that matches <*target name*> Hierarchy is specified using regular expression.

Pin Object

SDC Collection	Description
[get_pins <*instance name*> | <*pin name*>]:	Collection of a design pin that matches <*target name*> Hierarchy is specified using regular expression.

The get_nets command, for instance, can used to define the target for a create_generated_clock command.

Wildcard support

Only one wildcard (*) is supported. Also the wildcard should be at the end or beginning of the object name string. For example:

ab* matches abc, ab, abdefg, etc.

*bc matches abc, bbc, debc, etc.

Following is not supported:

a*b or a*b*c

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

create_clock

Creates a clock and defines its characteristics.

Note

In LSE timing, interclock domain paths are always blocked for create_clock. However, the interclock domain path is still valid for constraints such as set_false_path and set_multicycle_path.

Syntax create_clock [-name *name*] -period *period_value* [-waveform {value1 value2}] *source_object*

Arguments -name *name*

The name string specifies the name of the clock. If this parameter is not given, the name of the source object is used as the name of the clock. Virtual clocks are currently not supported.

-period *period_value*

This value is required and it specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The value specified for the period must be positive as the period of a clock must be greater than zero. The duty cycle of the clock is 50 percent.

-waveform {value1 value2}

The values are a list of edge values. Only two edges are supported. Floating values are accepted. Value1 must be less than value2, and the difference must be less than the clock period.

source_object

The source object is the object on which the clock constraint is defined. The source object can be a port object or a net object in the design. The object is obtained by using one of the get_ports or get_nets commands. If you specify a clock constraint on a source object that already has a clock, the new clock replaces the existing one. Only one source object is accepted. Wildcards are accepted as long as the resolution shows one port or net object.

Example The following example creates two clocks on ports CK1 and CK2 with a period of 6:

```
create_clock -name my_user_clock -period 6 [get_ports CK1]
create_clock -name my_other_user_clock -period 6 [get_ports CK2]
```

Example The following example creates a clock on port CK3 with a period of 7.1, and has two edges at 0 and 4.1:

```
create_clock -period 7.1 -waveform {0 4.1} [get_ports CK3]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

create_generated_clock

Creates an internally generated clock and defines its characteristics. This command is used when the clock being created is related to another clock. The generated clock will now be considered a clock when defining constraints such as input_delay.

Syntax create_generated_clock -source *reference_object* [-*master_clock clock_object*] [-divide_by *factor*] [-multiply_by *factor*] [-duty_cycle *value*] net_object

Arguments -source *reference_object*

The reference object is an object on which the source clock of the generated clock is defined. The source object can be a net object or a port object. The period of the generated clock is derived from the clock on the reference object using the multiply and divide factors.

-master_clock *clock_object*

If the master is defined, the master clock object becomes the source clock for the generated clock. This is an optional object used to identify a specific clock, if there is more than one clock on the source object.

-divide_by *factor*

This factor is the frequency division factor. The frequency of the generated clock is equal to the frequency of the source clock divided by this factor, if the multiply by factor is not specified. For instance, if this factor is equal to 2, the generated clock period is twice the reference clock period. Default value is 1.

-multiply_by *factor*

This factor specifies the frequency multiplication number to be used when finding the generated clock frequency. For instance, if the factor is equal to 2, the generated clock period is half the reference clock period. If both multiply_by and divide_by factors are used, the frequency is obtained by using both factors. Default value is 1.

-duty_cycle *value*

This value specifies the duty cycle in percentage of the clock period. The value can be floating point and ranges from 0 to 100. The default value is 50.

net_object

The net_object specifies the source of the clock constraint. This is usually an internal -net of the design. If you specify a clock constraint on a net that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one net.

This command creates a generated clock in the current design at a declared net_object by defining its frequency with respect to the frequency at the reference object. The static timing analysis tool uses this information to compute and propagate the generated clock's waveform across the clock network to the clock pins of all sequential elements driven by this target

Examples The following example creates a generated clock on pin pll1/CLKOP with a period twice as long as the period at the reference port CLK:

```
create_generated_clock -divide_by 2 -source [get_ports CLK]
[get_pins pll1/CLKOP]
```

The following example creates a generated clock at the primary output of myPLL with a period 3/4 of the period at the reference pin clk:

```
create_generated_clock -divide_by 3 -multiply_by 4  -source
[get_ports clk]  [get_pins myPLL/CLK1]
```

The following example shows a clock with a duty cycle of 60 percent:

```
create_generated_clock -duty_cycle 60 -source [get_ports clk]
[get_pins myPLL/CLK1]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

set_clock_groups

Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during timing analysis.

Syntax set_clock_groups *-asynchronous* | *-exclusive* -group *clock_objects* [-group *clock_objects*]

Arguments -asynchronous

Specifies that the clock groups are asynchronous to each other (meanwhile, Lattice assume all clocks are asynchronous). Two clocks are asynchronous with respect to each other if they have no phase relationship at all.

-exclusive

Specifies that clocks are mutually exclusive. Only one clock group will be active at any given time.

-group *clock_object*

Specifies the clock objects in a group. If you specify only one group, it means that the clocks in that group are exclusive or asynchronous with all other clocks in the design. A default other group is created for this single group. Whenever a new clock is created, it is automatically included in this group.

Examples The following example specifies two clock ports (clka and clkb) are asynchronous to each other.

```
create_clock -period 10.000 -name clka_port [get_ports clka]
create_clock -period 10.000 -name clkb_port [get_ports clkb]
# Set clka_port and clkb_port to be mutually exclusive clocks.
set_clock_groups -asynchronous -group [get_clocks clka_port] -
group [get_clocks clkb_port]
# The previous line is equivalent to the following two
commands.
set_false_path -from [get_clocks clka_port] -to [get_clocks
clkb_port]
set_false_path -from [get_clocks clkb_port] -to [get_clocks
clka_port]
```

The following example specifies four clock constraints that won't be active at the same time:

```
create_clock -period 10.000 -name clka_port [get_ports clka]
create_clock -period 10.000 -name clkb_port [get_ports clkb]
create_clock -period 10.000 -name clkc_port [get_ports clkc]
set_clock_groups -exclusive -group [get_clocks {clka_port
clkb_port}] -group [get_clocks clkc_port]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

set_false_path

Identifies paths that are considered false and excluded from timing analysis.

Syntax set_false_path [-from *from port_object or cell_object*] [-to *to port_object or cell_object*]

or

set_false_path [-through *through_net_object*]

Arguments -from *from port_object or cell_object*

Specifies the timing path start point. A valid timing starting point is a clock, a primary input, a combinational logic cell, or a sequential cell (clock-pin).

-to *to port_object or cell_object*

Specifies the timing path end point. A valid timing end point is a primary output, a combinational logic cell, or a sequential cell (data-pin).

-through *through_net_object*

Specifies a net through which the paths should be blocked.

Examples The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

The following example specifies all paths through the net UO/sigA as false:

```
set_false_path -from [get_ports clk1] -to [get_cells reg_2]
```

```
set_false_path -through [get_nets UO/sigA]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

set_input_delay

Defines the arrival time of an input relative to a clock.

Syntax set_input_delay *delay_value* [-max |-min] -clock *clock_object* *input_port_object*

Arguments *delay_value*

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

-max

Specifies that the delay value is the maximum delay.

-min

Specifies that the delay value is the minimum delay.

-clock *clock_object*

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument.

input_port_object

Provides one or more input ports in the current design to which delay_value is assigned. You can also use the keyword "all_inputs" to include all input ports.

Example The following example sets an input delay of 1.2 ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

Example The following example sets an input delay of 1.2 ns minimum and 1.5 ns maximum for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -min -clock [get_clocks CLK1] [get_ports
data1]
set_input_delay 1.5 -max -clock [get_clocks CLK1] [get_ports
data1]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

set_max_delay

Specifies the maximum delay for the timing paths.

Syntax set_max_delay *delay_value* [-from *from port_object or cell_object*] [-to *to port_object or cell_object*]

Arguments *delay_value*

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

If the path ending point is on a sequential device, the tool includes library setup time in the computed delay.

-from *from port_object or cell_object*

Specifies the timing path start point. A valid timing start point is a clock, a primary input, a combinational logic cell, or a sequential cell (clock pin).

-to *to port_object or cell_object*

Specifies the timing path end point. A valid timing end point is a primary output, a combinational logic cell, or a sequential cell (data pin)

Examples The following example sets a maximum delay by constraining all paths from ff1a:CLK to ff2e:D with a delay less than or equal to 5 ns:

```
set_max_delay 5 -from [get_cells ff1a] -to [get_cells ff2e]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

set_min_delay

Specifies the minimum delay for the timing paths.

Syntax set_min_delay *delay_value* [-from *from port_object or cell_object*] [-to *to port_object or cell_object*]

Arguments *delay_value*

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

If the path ending point is on a sequential device, the tool includes library hold time in the computed delay.

-from *from port_object or cell_object*

Specifies the timing path start point. A valid timing start point is a clock, a primary input, a combinational logic cell, or a sequential cell (clock pin).

-to *to port_object or cell_object*

Specifies the timing path end point. A valid timing end point is a primary output, a combinational logic cell, or a sequential cell (data pin)

Examples The following example sets a minimum delay by constraining all paths from ff1a:CLK to ff2e:D with a delay greater than or equal to 5 ns:

```
set_min_delay 5 -from [get_cells ff1a] -to [get_cells ff2e]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

set_multicycle_path

Defines a path that takes multiple clock cycles.

Syntax set_multicycle_path *ncycles* [-from *from net_object or cell_object*] [-to *to net_object or cell_object*]

Arguments *ncycles*

Specifies a value that represents the number of cycles the data path must have for setup check. The value is relative to the ending point clock and is defined as the delay required for arrival at the ending point.

-from *from net_object or cell_object*

Specifies the timing path start point. A valid timing start point is a sequential cell (clock pin) or a clock net (signal). You can also use the keyword "all_registers" to include all registers' clock inputs.

-to *to net_object or cell_object*

Specifies the timing path end point. A valid timing end point is a sequential cell (data-pin) or a clock-net (signal). You can also use the keyword "all_registers" to include all registers' data inputs.

Example The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_cells reg1] -to [get_cells
reg2]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

set_output_delay

Defines the output delay of an output relative to a clock.

Syntax set_output_delay *delay_value* [-max |-min] -clock *clock_object output_port_object*

Arguments *delay_value*

Specifies the amount of time from a reference clock to a primary output port.

-max

Specifies that the delay value is the maximum delay.

-min

Specifies that the delay value is the minimum delay.

-clock *clock_object*

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument.

output_port_object

Provides one or more (by wildcard) output ports in the current design to which delay_value is assigned. You can also use the keyword "all_outputs" to include all output ports.

Example The following example sets an output delay of 1.2 ns for all outputs relative to clki_c:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
set_output_delay 1.2 -clock [get_clocks CLK1] [all_outputs]
```

Example The following example sets an output delay of 1.2 ns minimum and 1.5 ns maximum for port data1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -min -clock [get_clocks CLK1] [get_ports
data1]
set_output delay 1.5 -max -clock [get_clocks CLK1] [get_ports
data1]
```

See Also ▶"Synopsys Design Constraints (SDC)" on page 1339

Lattice Synthesis Engine-Supported HDL Attributes

This section describes the Synplify Lattice Attributes that are supported by the Lattice Synthesis Engine (LSE). These attributes are directly interpreted by the engine and influence the optimization or structure of the output netlist. Traditional HDL attributes, such as UGROUP, are also compatible with LSE and are passed into the netlist to direct Map and Place & Route.

All HDL attributes have priority over Strategy settings.

See Also ▶"black_box_pad_pin" on page 1351

- “full_case” on page 1353
- “GSR” on page 1354
- “loc” on page 1356
- “parallel_case” on page 1357
- “syn_black_box” on page 1359
- “syn_encoding” on page 1360
- “syn_force_pads” on page 1365
- “syn_hier” on page 1367
- “syn_insert_pad” on page 1368
- “syn_keep” on page 1370
- “syn_maxfan” on page 1372
- “syn_multstyle” on page 1373
- “syn_noprune” on page 1375
- “syn_pipeline” on page 1377
- “syn_preserve” on page 1379
- “syn_ramstyle” on page 1381
- “syn_replicate” on page 1383
- “syn_romstyle” on page 1385
- “syn_srlstyle” on page 1386
- “syn_sharing” on page 1389
- “syn_state_machine” on page 1391
- “syn_use_carry_chain” on page 1395
- “syn_useenables” on page 1396
- “syn_useioff” on page 1398
- “translate_off/translate_on” on page 1399

black_box_pad_pin

This attribute specifies pins on a user-defined black box module. The pins are defined as I/O pads that are visible outside of the black box. If there is more than one port that is an I/O pad, list the ports inside double-quotes ("), separated by commas (,), and without enclosed spaces. This attribute must be used in conjunction with the syn_black_box attribute.

Verilog Syntax object /* synthesis syn_black_box black_box_pad_pin = "portList" */ ;

where object is a module declaration, and portList is a spaceless, comma-separated list of the black box port names that are I/O pads.

Verilog Example

```
module black_box_pad_pin2(
                    input[4:0] in1,
                    input[4:0] in2,
                    input clk,
                    output[4:0] q
                    )/*          synthesis          syn_black_box
                    black_box_pad_pin="in1(4:0),q" */;

  reg [4:0] q;
  always @(posedge clk)
    begin
       q <= in1 + in2;
    end
endmodule

module black_box_pad_pin_instan(
  input[4:0] in1,
  input[4:0] in2,
  input[4:0] in3,
  input clk,
  output[5:0] q_out
  );

  wire [4:0] q;
  reg [5:0] q_out;
  black_box_pad_pin2 test_123(
                           .in1(in1),
                           .in2(in2),
                           .clk(clk),
                           .q(q)
                           );

  always @(posedge clk)
    begin
       q_out <= q + in3;
    end
endmodule
```

VHDL Syntax attribute black_box_pad_pin of object : architecture is "portList" ;

where object is the architecture name of a black box. Data type is string. The portList is a spaceless, comma-separated list of the black box port names that are I/O pads.

VHDL Example

```
entity BBDLHS is
  port (D: in std_logic;
    E: in std_logic;
    GIN : in std_logic_vector(2 downto 0);
    Q : out std_logic );
end;

architecture BBDLHS_behav of BBDLHS is
end bl_box_behav;
attribute syn_black_box : boolean;
attribute syn_black_box of BBDLHS_behav : architecture is true;
attribute black_box_pad_pin : string;
attribute black_box_pad_pin of BBDLHS_behav : architecture is
"GIN(2:0),Q";
```

full_case

Directive. For Verilog designs only. When used with a case, casex, or casez statement, this directive indicates that all possible values have been given, and that no additional hardware is needed to preserve signal values.

Verilog Syntax object /* synthesis full_case */

Verilog Example

```
module full_case1 (q, in1, in2, in3, in4, sel);
  output q;
  input in1, in2, in3, in4;
  input [3:0] sel;
  reg q;
  always @(sel or in1 or in2 or in3 or in4)
    begin
      casez (sel) /* synthesis full_case */
        4'b11??: q = in4;
        4'b?1??: q = in3;
        4'b???1: q = in1;
        4'b??1?: q = in2;
        default: q = 'bx;
      endcase
    end
endmodule
```

GSR

This attribute specifies the use of the global set/reset routing resources. Allows the user to specify which portions of the design are to be altered in the way they respond to the GSR reset signal. Unless specified otherwise, all design elements will respond to the global reset signal if it is present.

Available values:

- ENABLED – This is the default value on most library elements. This value allows the software to determine the final value and will be overridden by the parent hierarchy if the parent has a value of anything other than ENABLED.
- DISABLED – This prevents the hierarchy or element from responding to the GSR value. It cannot be changed by the parent's value.
- FORCEENABLE – This forces the hierarchy or element to respond to the GSR value. It cannot be changed by the parent's value.
- IPENABLE – This forces the hierarchy or element to respond to the GSR value when IP is being used in evaluation mode. It cannot be changed by the parent's value. This value is only for internal Lattice IP to use. It should never be used within a design itself.

Verilog Syntax object /* synthesis GSR = {ENABLED | DISABLED | FORCEENABLE | IPENABLE} */ ;

Verilog Example

```
`timescale 1 ns / 1 ns
module top (reg_q, rotate_q, a, b, r_l, clk, rst)/* synthesis
GSR = "ENABLED" */;
output [7:0] reg_q, rotate_q;
input [7:0] a, b;
input clk, rst, r_l;

sub1 reg8_1 (.clk(clk), .data(a), .q(reg_q), .rst(rst));
sub2 rotate_1 (rotate_q, b, clk, r_l, rst);
endmodule
```

VHDL Syntax attribute gsr of object : objectType is ENABLED | DISABLED | FORCEENABLE | IPENABLE;

VHDL Example

```
architecture archtest of test is
attribute gsr : string;
attribute gsr of archtest : architecture is “ENABLED”;
```

loc

The loc attribute specifies pin locations for Lattice I/Os, instances, and registers, and forward-annotates them to the place-and-route tool. Refer to the Lattice databook for valid pin location values. If the attribute is on a bus, the software writes out bit-blasted constraints for forward-annotation.

Verilog Syntax object /* synthesis loc = "pinLocations" */ ;

In the syntax, pin_locations is a spaceless, comma-separated list of pin locations, starting with the lowest-indexed port and read left-to-right.

Verilog Example

```
I/O pin location
input [3:0]DATA0 /* synthesis loc="p10,p12,p11,p15" *;
Register pin location
reg data_in_ch1_buf_reg3 /* synthesis loc="R40C47" */;
Vectored internal bus
reg [3:0] data_in_ch1_reg /*synthesis loc =
"R40C47,R40C46,R40C45,R40C44" */;
```

VHDL Syntax attribute loc of object : objectType is "pinLocations" ;

In the syntax, pinLocations is a spaceless, comma-separated list of pin locations, starting with the lowest-indexed port and read left-to-right.

VHDL Example

```
entity mycomp is port(DATA0 : in std_logic_vector (3 downto 0);
                .
                .
                .
                );
attribute loc : string;
attribute loc of DATA0 : signal is "p10,p12,p11,p15";
```

parallel_case

Directive. For Verilog designs only. Forces a parallel-multiplexed structure rather than a priority-encoded structure. This is useful because case statements are defined to work in priority order, executing (only) the first statement with a tag that matches the select value.

If the select bus is driven from outside the current module, the current module has no information about the legal values of select, and the software must create a chain of disabling logic so that a match on a statement tag disables all following statements. However, if you know the legal values of select, you can eliminate extra priority-encoding logic with the parallel_case directive. In the following example, the only legal values of select are 4'b1000, 4'b0100, 4'b0010, and 4'b0001, and only one of the tags can be matched at a time. Specify the parallel_case directive so that tag-matching logic can be parallel and independent, instead of chained.

Note

Designers should be aware that it is possible for the priority of overlapping cases in post-synthesis simulation to mismatch with the priority behavior in RTL simulation when using this pragma.

Verilog Syntax You specify the directive as a comment immediately following the select value of the case statement.

object /* synthesis parallel_case */

where object is a case, casex or casez statement declaration.

Verilog Example

```
module parallel_case1 (q, in1, in2, in3, in4, sel);
  output q;
  input in1, in2, in3, in4;
  input [3:0] sel;
  reg q;
  always @(sel or in1 or in2 or in3 or in4)
    begin
      casez (sel) /* synthesis parallel_case */
        4'b11??: q = in4;
        4'b?1??: q = in3;
        4'b???1: q = in1;
        4'b??1?: q = in2;
        default: q = 'bx;
      endcase
    end
endmodule
```

If the select bus is decoded within the same module as the case statement, the parallelism of the tag matching is determined automatically, and the parallel_case directive is unnecessary.

syn_black_box

This attribute specifies that a Verilog module or VHDL architecture declaration is for a black box. Only the module's interface is defined for synthesis. The contents of a black box cannot be optimized during synthesis. A module can be a black box whether it is empty or not. However, the syn_black_box attribute cannot be used with the top-level module or architecture of a design. Additionally, the syn_black_box attribute is not supported for instances in Verilog or components in VHDL.

This attribute has an implicit Boolean value of 1 or true.

If any of the ports are I/O pads, add the black_box_pad_pin attribute. See "black_box_pad_pin" on page 1351.

Verilog Syntax object /* synthesis syn_black_box */ ;

where *object* is a module declaration.

Verilog Example

```
module bl_box(out,data,clk) /* synthesis syn_black_box */;
```

VHDL Syntax attribute syn_black_box of object : architecture is true ;

where *object* is an architecture name. Data type is Boolean.

VHDL Example

```
entity bl_box is
  port (data : in std_logic_vector (7 downto 0);
    clk : in std_logic;
    out : out std_logic);
end;

architecture bl_box_behav of bl_box is
end bl_box_behav;
attribute syn_black_box : boolean;
attribute syn_black_box of bl_box_behav : architecture is true;
```

See Also ▶"Creating Your Own Black Box Modules" on page 206

syn_encoding

This attribute specifies the encoding style for a finite state machine (FSM), overriding the default LSE encoding. The default encoding is based on the number of states in the FSM. This attribute takes effect only when LSE infers an FSM. This attribute has no effect when syn_state_machine is 0, which blocks inference of an FSM.

Values for syn_encoding are as follows:

- sequential – More than one bit of the state register can change at a time, but because more than one bit can be hot, the value must be decoded to determine the state. For example: 000, 001, 010, 011, 100
- onehot – Only two bits of the state register change (one goes to 0; one goes to 1) and only one of the state registers is hot (driven by a 1) at a time. For example: 0001, 0010, 0100, 1000
- gray – Only one bit of the state register changes at a time, but because more than one bit can be hot, the value must be decoded to determine the state. For example: 000, 001, 011, 010, 110

  There can be no more than four states for gray encoding. If the FSM has more than four states, LSE switches to sequential encoding.
- safe – If the state machine enters an invalid state, additional logic will drive the state machine into its reset state. The design must have a defined reset state.

  Safe encoding can be combined with either sequential or onehot encoding (not with gray encoding) as in:

  ```
  syn_encoding = "safe,onehot"
  ```

  If the safe value is given by itself, it combines with the encoding method of a preceding syn_encoding statement or the default method.

Verilog Syntax Object /* synthesis syn_encoding = "value" */;

Where object is an enumerated type and value is from the list above.

Verilog Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity syn_state_machine2 is
  port(
    clk : in std_logic;
    reset: in std_logic;
    en   : in std_logic;
    q  : out std_logic_vector(1 downto 0)
    );

end entity;

architecture behave of syn_state_machine2 is
  type state_type is(state0,state1,state2,state3);
  signal state,next_state:state_type;
  attribute syn_state_machine : boolean;
  attribute syn_state_machine of behave : architecture is true;
  attribute syn_encoding : string;
  attribute syn_encoding of state,next_state: signal is
"binary";
begin
  process(clk,reset)
  begin
    if reset = '1' then
      state <= state0;
    elsif clk'event and clk = '1' then
      state <= next_state;
    end if;
  end process;
  process(state)
  begin
    case state is
      when state0 =>
        if (en = '1') then
           q <= "00";
        end if;
        next_state <= state1;
      when state1=>
        if (en = '1') then
           q <= "01";
        end if;
        next_state <= state2;
      when state2 =>
        if (en = '1') then
           q <= "10";
        end if;
        next_state <= state3;
      when state3 =>
        if (en = '1') then
           q <= "11";
        end if;
        next_state <= state0;
    end case;
  end process;
end behave
```

VHDL Syntax attribute syn_encoding of object: objectType is "value";

Where object is an enumerated type and value is from the list above.

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity syn_encoding1 is
  port(
    clk : in std_logic;
    reset: in std_logic;
    en   : in std_logic;
    q  : out std_logic_vector(1 downto 0)
    );
end entity;

architecture behave of syn_encoding1 is
  signal state : std_logic_vector(3 downto 0);
  constant state0 : std_logic_vector(3 downto 0) := "1000";
  constant state1 : std_logic_vector(3 downto 0) := "0100";
  constant state2 : std_logic_vector(3 downto 0) := "0010";
  constant state3 : std_logic_vector(3 downto 0) := "0001";
  attribute syn_encoding : string;
  attribute syn_encoding of state : signal is "safe,onehot";
begin
  process(clk,reset,en)
  begin
    if reset = '1' then
      state <= state0;
      q <= "00";
    elsif clk'event and clk = '1' then
      case state is
        when state0 =>
          if (en = '1') then
             q <= "00";
          end if;
          state <= state1;
        when state1=>
          if (en = '1') then
             q <= "01";
          end if;
          state <= state2;
        when state2 =>
          if (en = '1') then
             q <= "10";
          end if;
          state <= state3;
        when state3 =>
          if (en = '1') then
             q <= "11";
          end if;
          state <= state0;
        when others => null;
      end case;
    end if;
  end process;
end behave
```

syn_force_pads

This attribute prevents unused ports from being optimized away to allow I/O pad insertion on the unused port. This attribute is not supported at the global level. Instead, set the use_io_insertion option to control I/O insertion globally.

This attribute is supported in the rtl, and it will override the global use_io_insertion option setting on the given input, output, or bidir port.

For example, in the following Verilog file, the syn_force_pads attribute can be set to 1 on an unused input port (dataz), and it will not be optimized away, regardless of the use_io_insertion global setting.

Verilog syntax object /* synthesis syn_force_pads = {1 | 0} */;

where object is port declaration.

Verilog Example

```
`define DSIZE 9
`define OSIZE 18

module multp9x9(dataout, dataax, dataay, dataz, clk, rst, ce);
   output [`OSIZE-1:0] dataout;
   input [`DSIZE:0] dataz /* synthesis syn_force_pads = 1*/;
   input [`DSIZE-1:0] dataax, dataay;
   input clk, rst, ce;
   reg  [`DSIZE-1:0] dataax_reg, dataay_reg;

   reg [`OSIZE-1:0] dataout;
   wire [`OSIZE-1:0] dataout_tmp ;
   assign dataout_tmp = dataax_reg * dataay_reg;

   always @(posedge clk or posedge rst)
   begin
     if (rst)
     begin
         dataax_reg <= 0;
         dataay_reg <= 0;
         dataout <= 0;
     end
     else if (ce == 1'b1)
     begin
         dataax_reg <= dataax;
         dataay_reg <= dataay;
         dataout <= dataout_tmp;
     end
   end
endmodule
```

To force I/O pads to be inserted for input ports that do not drive logic, follow the guidelines below.

- To force I/O pad insertion on an individual port, set the syn_force_pads attribute on the port with a value to 1. To disable I/O insertion for a port, set the attribute on the port with a value of 0.

Enable this attribute to preserve user-instantiated pads, insert pads on unconnected ports, insert bi-directional pads on bi-directional ports instead of converting them to input ports, or insert output pads on unconnected outputs.

If you do not set the syn_force_pads attribute, the synthesis design optimizes any unconnected I/O buffers away.

VHDL syntax Attribute syn_force_pads of object: objectType is "true | false" ;

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity multp9x9 is
  port ( dataout : out std_logic_vector(17 downto 0);
      dataax, dataay: in std_logic_vector( 8 downto 0);
           dataz : in std_logic_vector(8 downto 0);
           clk,rst,ce: in std_logic
    );
  attribute syn_force_pads : string;
  attribute syn_force_pads of dataz : signal is "true";

end multp9x9;

architecture rtl of multp9x9 is

signal dataax_reg,dataay_reg: std_logic_vector(8 downto 0);
signal dataout_tmp: std_logic_vector(17 downto 0);

begin

  dataout_tmp <= dataax_reg * dataay_reg;
  process (clk, rst)
  begin
      if rst = '1' then
           dataax_reg <= (others => '0');
           dataay_reg <= (others => '0');
           dataout <= (others => '0');
      elsif clk'event and clk = '1' then
        if ce = '1' then
           dataax_reg <= dataax;
           dataay_reg <= dataay;
           dataout <= dataout_tmp;
      end if;
      end if;
  end process;

end rtl;
```

syn_hier

This attribute allows you to control the amount of hierarchical transformation that occurs across boundaries on module or component instances during optimization. This attribute cannot be applied globally. The user must set this attribute on the selective modules to stop cross-boundary optimizations.

syn_hier Values The following value can be used for syn_hier:

hard – Preserves the interface of the design unit with no exceptions. This attribute affects only the specified design units.

Verilog Syntax object /* synthesis syn_hier = "value" */ ;

where *object* can be a module declaration and value can be any of the values described in syn_hier Values. Check the attribute values to determine where to attach the attribute.

Verilog Example

```
module top1 (Q, CLK, RST, LD, CE, D)
   /* synthesis syn_hier = "hard" */;
```

VHDL Syntax attribute syn_hier of object : architecture is "value" ;

where *object* is an architecture name and value can be any of the values described in syn_hier Values. Check the attribute values to determine the level at which to attach the attribute.

VHDL Example

```
architecture struct of cpu is
attribute syn_hier : string;
attribute syn_hier of struct: architecture is "hard";
```

syn_insert_pad

This attribute removes an existing I/O buffer from a port or net when I/O buffer insertion is enabled.

The syn_insert_pad attribute is used when the use_io_insertion global option is enabled (when I/O buffers are automatically inserted) to allow users to selectively remove an individual buffer from a port or net.

It can also be used to force an I/O buffer to be inserted on a specific port or net, if the use_io_insertion global option is disabled.

- Setting the attribute to 0 on a port or net removes the I/O buffer (or prevents an I/O buffer from being automatically inserted, if the use_io_insertion global option is enabled).
- Setting the attribute to 1 on a port or net forces an I/O buffer to be inserted if the use_io_insertion global option is disabled.

Verilog Syntax object /* synthesis syn_insert_pad = {1 | 0} */;

where *object* is a port or net declaration.

Verilog Example

```
`define OSIZE 16
`define DSIZE 8

module mac8x8 (dataout, x, y, clk, rst);
 output [`OSIZE:0] dataout;
 input [`DSIZE-1:0] x, y;
 input clk;
 input rst /* synthesis syn_insert_pad = 0 */;
 reg  [`OSIZE:0] dataout;
 reg [`DSIZE-1:0] x_reg, y_reg;
 wire [`OSIZE-1:0] multout ;
 wire [`OSIZE:0] sum_out;

 assign multout = x_reg * y_reg;
 assign sum_out = multout + dataout;

 always @(posedge clk or posedge rst)
 begin
   if (rst)
   begin
      x_reg = 0;
      y_reg = 0;
      dataout = 0;
   end
   else
   begin
      x_reg = x;
      y_reg = y;
      dataout = sum_out;
   end
 end
endmodule
```

In the previous example, the input port labeled “rst” will not have an input buffer connected to it in the technology-mapped netlist after LSE completes.

VHDL Syntax attribute syn_insert_pad of object : objectType is "true | false" ;

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;

entity register_en_reset is
          generic (
                    width : integer := 8
                     );
  port (
    datain  : in std_logic_vector(width-1 downto 0);
    clk     : in std_logic;
    enable  : in std_logic;
    reset   : in std_logic;
    dataout : out std_logic_vector(width-1 downto 0)
    );
  attribute syn_insert_pad : string;
  attribute syn_insert_pad of reset : signal is "false";


end register_en_reset;

architecture lattice_behav of register_en_reset is

begin
    process (clk,reset)
    begin
      if (reset = '1') then
                      dataout <= (others => '0');
      elsif (rising_edge(clk) and enable = '1') then
        dataout <= datain;
      end if;
    end process;
end lattice_behav;
```

syn_keep

This attribute keeps the specified net intact during optimization and synthesis.

Verilog Syntax object /* synthesis syn_keep = 1 */ ;

where *object* is a wire or reg declaration. Make sure that there is a space between the object name and the beginning of the comment slash (/).

Verilog Example

```
module syn_keep1(
        input a,
        input b,
        input clk,
        output q1,
        output q2);
  reg temp1;
  reg temp2;
  reg q1;
  reg q2;
  wire or_result;
  wire keep1/* synthesis syn_keep=1 */;
  wire keep2/* synthesis syn_keep=1 */;

  always @(posedge clk)
    begin
      temp1 = a;
      temp2 = b;
    end
  assign or_result = (temp1 | temp2);
  assign keep1 = or_result;
  assign keep2 = or_result;
  always@(posedge clk)
    begin
      q1 = keep1;
      q2 = keep2;

    end
endmodule
```

VHDL Syntax attribute syn_keep of object : objectType is true ;

where *object* is a single or multiple-bit signal.

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;

entity syn_keep1 is
  port(
    a : in std_logic;
    b : in std_logic;
    clk: in std_logic;
    q1: out std_logic;
    q2: out std_logic
    );
end entity;

architecture behave of syn_keep1 is
  signal temp1 : std_logic;
  signal temp2 : std_logic;
  signal keep1 : std_logic;
  signal keep2 : std_logic;
  signal or_result : std_logic;
  attribute syn_keep: boolean;
  attribute syn_keep of keep1,keep2: signal is true;
begin
  process(clk)
  begin
    if clk'event and clk = '1' then
      temp1 <= a;
      temp2 <= b;
    end if;
  end process;

  or_result <= (temp1 or temp2);
  keep1 <= or_result;
  keep2 <= or_result;

  process(clk)
  begin
    if clk'event and clk = '1' then
      q1 <= keep1;
      q2 <= keep2;
    end if;
  end process;

end behave;
```

syn_maxfan

This attribute overrides the default (global) fan-out guide for an individual input port, net, or register output.

Verilog Syntax object /* synthesis syn_maxfan = "value" */ ;

Note

LSE will take integer values for non-integral values to syn_maxfan attribute.

For example, syn_maxfan value of 5.1 will be truncated to 5.

Verilog Example

```
module test (registered_data_out, clock, data_in);
output [31:0] registered_data_out;
input clock;
input [31:0] data_in /* synthesis syn_maxfan=1000 */;
reg [31:0] registered_data_out /* synthesis syn_maxfan=1000 */;
```

VHDL Syntax attribute syn_maxfan of object : objectType is "value" ;

VHDL Example

```
entity test is
   port (clock : in bit;
         data_in : in bit_vector(31 downto 0);
         registered_data_out: out bit_vector(31 downto 0) );
attribute syn_maxfan : integer;
attribute syn_maxfan of data_in : signal is 1000;
```

See Also “Optimizing LSE for Area and Speed” on page 581

syn_multstyle

This attribute specifies whether the multipliers are implemented as dedicated hardware blocks or as logic.

syn_multstyle Values block_mult | logic

Value Description Default block_mult Implements the multipliers as dedicated hardware blocks (Lattice: DSP blocks)

This attribute only applies to families that use DSP blocks on the device. To override this behavior, specify a value of logic.

Verilog Syntax input net /* synthesis syn_multstyle = "block_mult | logic" */;

Verilog Example

```
module syn_multstyle1(
            input [7:0] in1,
            input [7:0] in2,
            input rst,
            input clk,
            output [15:0] result);

  reg [7:0] temp1,temp2;
  reg [15:0] result;
  wire [15:0] product /*synthesis syn_multstyle = "logic"*/;

  always@(posedge clk ,negedge rst)
    begin
      if(!rst)
        begin
          temp1 = 'b0;
          temp2 = 'b0;
        end
      else
        begin
          temp1  = in1;
          temp2  = in2;
        end
    end

  assign product = temp1*temp2;

  always@(posedge clk, negedge rst)
    begin
      if(!rst)
        begin
          result  = 'b0;
        end
      else
        begin
          result = product;
        end
      end
endmodule
```

VHDL Syntax attribute syn_multstyle of instance : signal is "block_mult | logic";

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity mac_ecp4_m5678_09x09_ir_or_bit_up is generic (widtha :
natural := 9;
         widthb : natural := 9);
port (
      ina     : in  std_logic_vector (0 to widtha - 1);
      inb_0   : in  std_logic;
      inb_1   : in  std_logic;
      inb_2   : in  std_logic;
      inb_3   : in  std_logic;
      inb_4   : in  std_logic;
      inb_5   : in  std_logic;
      inb_6   : in  std_logic;
      inb_7   : in  std_logic;
      inb_8   : in  std_logic;
      clk     : in  std_logic;
      mout    : out  std_logic_vector (0 to widtha+widthb -
1));
    attribute syn_multstyle : string;
  attribute syn_multstyle of mout: signal is "block_mult"; end
mac_ecp4_m5678_09x09_ir_or_bit_up;

architecture rtl of mac_ecp4_m5678_09x09_ir_or_bit_up is
signal mout_s     : std_logic_vector (0 to widtha+widthb - 1);
signal reg1_ina : std_logic_vector(0 to widtha - 1); signal
reg1_inb : std_logic_vector(0 to widthb - 1); signal reg_mout :
std_logic_vector(0 to widtha+widthb - 1);
signal inb     : std_logic_vector (0 to widthb - 1);
begin

inb <= inb_8 & inb_7 & inb_6 & inb_5 & inb_4 & inb_3 & inb_2 &
inb_1 & inb_0 ;

 reg_mout <= reg1_ina * reg1_inb+mout_s;
  mout <= mout_s;

process (clk) begin
  if
   rising_edge (clk) then
      reg1_ina <= ina;
      reg1_inb <= inb;
      mout_s <= reg_mout;
  end if;
end process;

end rtl;
```

syn_noprune

This attribute prevents instance optimization for black-box modules (including technology-specific primitives) with unused output ports. This attribute is not a global attribute. It works on the component basis. The user must set the attribute on the instance.

Verilog Syntax object /* synthesis syn_noprune = 1 */ ;

where object is a module an instance. The data type is Boolean.

Verilog Example

```
module top(a1,b1,c1,d1,y1,clk);
output y1;
input a1,b1,c1,d1;
input clk;
wire x2,y2;
reg y1;
syn_noprune u1(a1,b1,c1,d1,x2,y2) /* synthesis syn_noprune=1 */
;

always @(posedge clk)
   y1<= a1;

endmodule
```

VHDL Syntax attribute syn_noprune of object : objectType is true ;

where the data type is boolean, and object is a component.

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
entity top is
   port (a1, b1 : in std_logic;
         c1,d1,clk : in std_logic;
         y1 :out std_logic );
end ;
architecture behave of top is
component noprune
port (a, b, c, d : in std_logic;
      x,y : out std_logic );
end component;
signal x2,y2 : std_logic;
attribute syn_noprune : boolean;
attribute syn_noprune of noprune : component is true;
begin
   u1: noprune port map(a1, b1, c1, d1, x2, y2);
   process begin
      wait until (clk = '1') and clk'event;
      y1 <= a1;
   end process;
end;
```

syn_pipeline

This attribute permits registers to be moved to improve timing. Specifies that registers that are outputs of Multipliers/Adders can be moved to improve timing. Depending on the criticality of the path, the tool moves the output register to the input side.

Verilog Syntax object /* synthesis syn_pipeline = {1 | 0} */ ;

where *object* is a register declaration.

The value of 0 (or false) indicates pipelining for the specified register is disabled, which means the register position in the design is fixed.

The value of 1 (or true) indicates pipelining for the specified register is allowed, which means the register may be moved if it helps improve timing.

LSE will identify registers that are candidates for possible pipelining based on running RTL timing analysis. It may identify some candidate registers, or it may determine there are none that are suitable.

If LSE decides no candidate registers for pipelining exist, if the user sets the syn_pipeline attribute to “1” on a specific register in the RTL to force pipelining for that register, that attribute will not be honored.

If global pipelining is enabled for a design, and given one or more registers that LSE has identified as possible candidates for pipelining, the user may prevent these registers from being pipelined by setting synthesis attribute syn_pipeline=0 for each of those registers in the RTL.

Verilog Example

```
module pipeline (a,b,c,d,clk,out);

input [3:0] a,b,c,d;
input clk;
output[7:0]out;

reg[7:0]out,out1 /* synthesis syn_pipeline = 0 */;
reg[3:0] a_temp,b_temp,c_temp,d_temp;

always @(posedge clk)
begin
   a_temp <= a;
   b_temp <= b;
   c_temp <= c;
   d_temp <= d;
   out1 <= (a_temp * b_temp) +(c_temp * d_temp);
   out <= out1;
end
endmodule
```

In the previous example, the registers labeled “out1” will not be moved to the input side of the adder to improve timing.

VHDL Syntax attribute syn_pipeline of object : objectType is {true|false} ;

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity syn_pipeline_exp is
port (CLK_0 : in std_logic;
    A_IN : in std_logic_vector(3 downto 0);
    B_IN : in std_logic_vector(3 downto 0);
    RST : out std_logic_vector(7 downto 0)
    );
end syn_pipeline_exp;

architecture rtl of syn_pipeline_exp is
signal A_REGSTR : std_logic_vector(3 downto 0);
signal B_REGSTR : std_logic_vector(3 downto 0);
signal TMP : std_logic_vector(7 downto 0);
signal TMP1 : std_logic_vector(7 downto 0);
signal TMP2 : std_logic_vector(7 downto 0);
attribute syn_pipeline : string;
attribute syn_pipeline of TMP1 : signal is "true";

begin
  process(CLK_0)
  begin
    if (CLK_0'event and CLK_0 = '1') then
      TMP <= A_REGSTR * B_REGSTR;
      A_REGSTR <= A_IN;
      B_REGSTR <= B_IN;
      TMP1 <= TMP;
      TMP2 <= TMP1;
      RST <= TMP2;
    end if;
  end process;

end rtl;
```

syn_preserve

This attribute prevents sequential optimizations such as constant propagation and inverter push-through from removing the specified register. The syn_encoding attribute is not honored if there is a syn_preserve attribute on any of the state machine registers.

Verilog Syntax object /* synthesis syn_preserve = 1 */ ;

where object is a register definition signal or a module.

Verilog Example

```
module syn_preserve1(
          input[3:0]in1,
          input[3:0]in2,
          input[3:0]in3,
          input clk,
          output [7:0] result,
          output [3:0] sum
          )/* synthesis syn_preserve= 1*/;

reg [7:0] result/*synthesis syn_multstyle = "EBR"*/;
reg [3:0] temp1,temp2,temp3;
reg [3:0] sum;

always@(posedge clk)
  begin
    temp1 = in1 & in2;
    temp2 = !temp1;
    temp3 = temp1 & temp2;
    result = temp3*in3;
    sum = temp3 + in3;
  end
endmodule
```

VHDL Syntax attribute syn_preserve of object : objectType is true ;

where object is an output port or an internal signal that holds the value of a state register or architecture.

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity syn_preserve2 is
  port(
     in1: in std_logic_vector(3 downto 0);
     in2: in std_logic_vector(3 downto 0);
     in3: in std_logic_vector(3 downto 0);
     clk: in std_logic;
     result: out std_logic_vector(7 downto 0);
     sum   : out std_logic_vector(3 downto 0)
    );
end entity;

architecture behave of syn_preserve2 is
  signal temp1,temp2,temp3 : std_logic_vector(3 downto 0);
  attribute syn_preserve : boolean;
  attribute syn_preserve of behave: architecture is true;
  attribute syn_multstyle : string;
  attribute syn_multstyle of result: signal is "EBR";
begin
  process(clk)
  begin
    if clk'event and clk = '1' then
      temp1 <= in1 and in2;
      temp2 <= not temp1;
      temp3 <= temp1 and temp2;
      result <= temp3*in3;
      sum <= temp3 + in3;
    end if;
  end process;
end behave;
```

syn_ramstyle

The syn_ramstyle attribute specifies the implementation to use for an inferred RAM. You apply syn_ramstyle globally, to a module, or to a RAM instance. To turn off RAM inference, set its value to registers.

The following values can be specified globally or on a module or RAM instance:

- registers – Causes an inferred RAM to be mapped to registers (flip-flops and logic) rather than the technology-specific RAM resources.
- distributed – Causes the RAM to be implemented using the distributed RAM or PFU resources.
- block_ram – Causes the RAM to be implemented using the dedicated RAM resources. If your RAM resources are limited, you can use this attribute to map additional RAMs to registers instead of the dedicated or distributed RAM resources.
- no_rw_check (some modes, but all technologies). – You cannot specify this value alone. Without no_rw_check, the synthesis tool inserts bypass logic around the RAM to prevent the mismatch. If you know your design does not read and write to the same address simultaneously, use no_rw_check to eliminate bypass logic. Use this value only when you cannot simultaneously read and write to the same RAM location and you want to minimize overhead logic.

Verilog Syntax object /* synthesis syn_ramstyle = "string" */ ;

where object is a register definition (reg) signal. The data type is string.

Verilog Example

```
module ram4 (datain,dataout,clk);
output [31:0] dataout;
input clk;
input [31:0] datain;
reg [7:0] dataout[31:0] /* synthesis syn_ramstyle="block_ram" */;
```

VHDL Syntax attribute syn_ramstyle of object : objectType is "string" ;

where object is a signal that defines a RAM or a label of a component instance. Data type is string.

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
entity ram4 is
   port (d : in std_logic_vector(7 downto 0);
         addr : in std_logic_vector(2 downto 0);
         we : in std_logic;
         clk : in std_logic;
         ram_out : out std_logic_vector(7 downto 0) );
end ram4;
library synplify;
architecture rtl of ram4 is
type mem_type is array (127 downto 0) of std_logic_vector (7
downto 0);
signal mem : mem_type; -- mem is the signal that defines the
RAM
attribute syn_ramstyle : string;
attribute syn_ramstyle of mem : signal is "block_ram";
```

syn_replicate

This attribute controls replication. The synthesis tool can automatically replicate registers during optimization. This attribute disables replication either globally or on a per register basis.

Verilog Syntax object /* synthesis syn_replicate = 1 | 0 */;

Verilog Example For example:

```
module syn_replicate1 (en1,en2,clk,in1,in2,q);
  input en1,en2;
  input clk;
  input [6:0]in1,in2;
  output [6:0]q;
  reg [6:0]q;
  reg enc /* synthesis syn_maxfan = 1 syn_replicate = 1*/;

  always @(posedge clk)
    begin
      enc = en1 & en2;
    end

  always @(posedge clk)
    begin
      if (enc)
        q = in1;

      else
        q = in2;
    end
endmodule
```

VHDL Syntax attribute syn_replicate of object : objectType is true | false ;

VHDL Example For example:

```
library ieee;
use ieee.std_logic_1164.all;

entity syn_replicate2 is
  port(
     en1: in std_logic;
     en2: in std_logic;
     clk: in std_logic;
     in1: in std_logic_vector(6 downto 0);
     in2: in std_logic_vector(6 downto 0);
     q:  out std_logic_vector(6 downto 0)
    );
end entity;

architecture behave of syn_replicate2 is
  signal enc : std_logic;
  attribute syn_maxfan: integer;
  attribute syn_maxfan of behave : architecture is 1;
  attribute syn_replicate: boolean;
  attribute syn_replicate of enc : signal is false;
begin
  process(clk)
  begin
    if clk'event and clk = '1' then
      enc <= (en1 and en2);
    end if;
  end process;

  process(clk)
  begin
    if enc = '1' then
      q <= in1;
    else
      q <= in2;
    end if;
  end process;
end behave;
```

See Also “Optimizing LSE for Area and Speed” on page 581

syn_romstyle

This attribute allows you to implement ROM architectures using dedicated or distributed ROM. Infer ROM architectures using a CASE statement in your code.

For the synthesis tool to implement a ROM, at least half of the available addresses in the CASE statement must be assigned a value. For example, consider a ROM with six address bits (64 unique addresses). The case statement for this ROM must specify values for at least 32 of the available addresses. You can apply the syn_romstyle attribute globally to the design by adding the attribute to the module or entity.

The following values can be specified globally on a module or ROM instance:

- auto – Allows the synthesis tool to chose the best implementation to meet the design requirements for speed, size, etc.
- logic – Causes the ROM to be implemented using the distributed ROM or PFU resources.
- EBR – Causes the ROM to be implemented using the dedicated ROM resources. If your ROM resources are limited, you can use this attribute to map additional ROM to registers instead of the dedicated or distributed RAM resources.

Verilog Syntax object /* syn_romstyle = "auto(default) | EBR | logic" */ ;

Verilog Example

```
reg [8:0] z /* synthesis syn_romstyle = "EBR" */;
```

VHDL Syntax attribute syn_romstyle of object : object_type is "auto(default) | EBR | logic" ;

VHDL Example

```
signal z : std_logic_vector(8 downto 0);
attribute syn_romstyle : string;
attribute syn_romstyle of z : signal is "logic";
```

syn_srlstyle

This attribute determines how to implement the sequential shift components.

Verilog Syntax object /* synthesis syn_srlstyle = "string",

where string can take one of the following values:

registers: seqShift register components are implemented as registers.

distributed: seqShift register components are implemented as distributed RAM.

block_ram: seqShift register components are implemented as block RAM

If the attribute value set by the user cannot be honored (for example, the user sets the attribute value to "block_ram", however, the selected device does not contain enough available EBR blocks to implement the shift register), LSE will display a message to indicate this.

" | registers | distributed | |block_ram" */ ;

In the above syntax, *object* is a register declaration.

Verilog Example The following example implements seqShift components as distributed memory with any required fabric logic.

```
module test_srl(clk, enable, dataIn, result, addr);
input clk, enable;
input [3:0] dataIn;
input [3:0] addr;
output [3:0] result;
reg [3:0] regBank[15:0]
  /* synthesis syn_srlstyle="distributed" */;
integer i;
always @(posedge clk) begin
  if (enable == 1) begin
    for (i=15; i>0; i=i-1) begin
      regBank[i] <= regBank[i-1];
    end
  regBank[0] <= dataIn;
  end
end
assign result = regBank[addr];
endmodule
```

The following example implements a seqShift for 16x256 bits wide and serial in and serial out register using syn_srlstyle set to block_ram.

VHDL Syntax attribute syn_srlstyle of object : signal is

" registers | distributed |block_ram " ;

In the above syntax, object is a register.

```
// shift left register with 16X256 bits width and serial in and
serial out
module test(clock, arst, sr_en, shiftin, shiftout);
parameter sh_len=16;
parameter sh_width=256;
parameter ARESET_VALUE = {(sh_width){1'b0}};
input clock,arst,sr_en;
input [sh_width-1:0] shiftin;
output [sh_width-1:0] shiftout;
integer i;
reg [sh_width-1:0] sreg [sh_len-1:0] /* synthesis
syn_srlstyle="block_ram" */;
  always @(posedge clock or posedge arst)
  begin
    if(arst)
      begin
        for(i = 0;i <= sh_len-1;i = i+1)
        sreg[i] <= ARESET_VALUE ;
      end
    else
  begin
    if(sr_en)
    begin
      sreg[0] <= shiftin;
      for(i=sh_len-1;i>0;i=i-1)
        sreg[i] <= sreg[i-1];
      end
    end
  end
assign shiftout = sreg[sh_len-1];
endmodule
```

Verilog Example The example below implements seqShift components as distributed memory primitives:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity d_p is
  port (clk : in std_logic;
    data_out : out std_logic_vector(127 downto 0));
  end d_p;

architecture rtl of d_p is
type dataAryType is array(3 downto 0) of
  std_logic_vector(127 downto 0);
signal h_data_pip_i : dataAryType;
attribute syn_srlstyle : string;
attribute syn_srlstyle of h_data_pip_i : signal
is "distributed";
begin
  process (Clk)
  begin
    if (Clk'Event And Clk = '1') then
      h_data_pip_i <= (h_data_pip_i(2 DOWNTO 0)) &
      h_data_pip_i(3);
    end if;
  end process;
data_out <= h_data_pip_i(0);
end rtl;
```

syn_sharing

Directive. Enables or disables the sharing of operator resources during the compilation stage of synthesis.

The syn_sharing directive controls resource sharing during the compilation stage of synthesis. This is a compiler-specific optimization that does not affect the mapper; this means that the mapper might still perform resource sharing optimizations to improve timing, even if syn_sharing is disabled.

If you disable resource sharing globally, you can use the syn_sharing directive to turn on resource sharing for specific modules or architectures.

Verilog Syntax object /* synthesis syn_sharing=”on | off” */ ;

Verilog Example

```
module syn_sharing1 (
          input [7:0] inA1,
          input [7:0] inA2,
          input [7:0] inB1,
          input [7:0] inB2,
          input clk,
          input sel1,
          input sel2,
          input rst,
          output [15:0] product1,
          output [15:0] product2
          )/*synthesis syn_sharing = 1*/;

  reg [15:0] product1,product2;
  wire [15:0] temp1,temp2;
  assign temp1 = inA1*inB1;
  assign temp2 = inA2*inB2;
  always@(posedge clk)
    begin
      if(sel1)
        begin
          if (sel2)
            product1 = temp1;
          else
            product1 = temp2;
        end
      else
        begin
          if (sel2)
            product2 = temp1;
          else
            product2 = temp2;
        end
    end
endmodule
```

VHDL Syntax attribute syn_sharing of object : objectType is “true | false” ;

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity syn_sharing2 is
  port(
    inA1 : in std_logic_vector(7 downto 0);
    inA2 : in std_logic_vector(7 downto 0);
    inB1 : in std_logic_vector(7 downto 0);
    inB2 : in std_logic_vector(7 downto 0);
    clk  : in std_logic;
    sel1 : in std_logic;
    sel2 : in std_logic;
    rst  : in std_logic;
    product1 : out std_logic_vector(15 downto 0);
    product2 : out std_logic_vector(15 downto 0)
    );
end entity;

architecture behave of syn_sharing2 is
 signal temp1,temp2: std_logic_vector(15 downto 0);
 attribute syn_sharing : boolean;
 attribute syn_sharing of behave : architecture is false;
begin
  temp1 <= inA1*inB1;
  temp2 <= inA2*inB2;
  process(clk)
  begin
    if clk'event and clk ='1' then
      if sel1 = '1' then
        if sel2 = '1' then
          product1 <= temp1;
        else
          product1 <= temp2;
        end if;
      else
        if sel2 = '1' then
          product2 <= temp1;
        else
          product2 <= temp2;
        end if;

      end if;
    end if;
  end process;
end behave;
```

syn_state_machine

This attribute enables/disables state-machine optimization on individual state registers in the design. To extract some state machines, use this attribute with a value of 1 on just those individual state-registers to be extracted. If there are state machines in your design that you do not want extracted, use syn_state_machine with a value of 0 to override extraction on just those individual state registers.

All state machines are usually detected during synthesis. However, on occasion there are cases in which certain state machines are not detected. You can use this attribute to declare those undetected registers as state machines.

The syn_sharing attribute only can be used in architecture. The syn_sharing attribute cannot be used in entity.

Verilog Syntax object /* synthesis syn_state_machine = 0 | 1 */ ;

where *object* is a state register. Data type is Boolean: 0 does not extract an FSM, 1 extracts an FSM.

Verilog Example

```
module syn_state_machine1 (clk, reset, en, q);
  input clk, reset, en;
  output[1:0]q;
  reg q;
  reg [3:0] state,next_state /* synthesis syn_state_machine = 0
*/;
  parameter state0 = 4'b1000;
  parameter state1 = 4'b0100;
  parameter state2 = 4'b0010;
  parameter state3 = 4'b0001;
  always @(posedge clk or posedge reset)
    begin
      if (reset)
          state <= state0;
      else
        state <= next_state;
    end

  always @(state)
    begin
      case (state)
        state0:
          begin
            if (en == 1)
              q <= 2'b00;
            next_state <= state1;
          end

        state1:
          begin
            if (en == 1)
              q <= 2'b01;
            next_state <= state2;
          end
        state2:
          begin
            if (en == 1)
              q <= 2'b10;
            next_state <= state3;
          end
        state3:
          begin
            if (en == 1)
              q <= 2'b11;
            next_state <= state0;
          end
      endcase
    end
endmodule
```

VHDL Syntax attribute syn_state_machine of object : objectType is true | false ;

where *object* is a signal that holds the value of the state machine.

VHDL Example

```
attribute syn_state_machine of current_state: signal is true;
```

Following is the source code used for the example in the previous figure.

```
library ieee;
use ieee.std_logic_1164.all;
entity syn_statemachine_exp is
port (CLK_0, RESET, IN1 : in std_logic;
                  OUT1 : out std_logic_vector (2 downto 0)
      );
end syn_statemachine_exp;

architecture behave of syn_statemachine_exp is
type ST_VALS is (STATE0, STATE1, STATE2, STATE3);
signal STATE, NXT_ST: ST_VALS;
attribute syn_state_machine : boolean;
attribute syn_state_machine of STATE : signal is true;

begin
  process (CLK_0, RESET)
  begin
    if RESET = '1' then
      STATE <= STATE0;
    elsif rising_edge(CLK_0) then
      STATE <= NXT_ST;
    end if;
  end process;

  process (STATE, IN1)
  begin
    case STATE is
      when STATE0 =>
        OUT1 <= "000";
        if IN1 = '1' then NXT_ST <= STATE1;
        else NXT_ST <= STATE0;
        end if;
      when STATE1 =>
        OUT1 <= "001";
        if IN1 = '1' then NXT_ST <= STATE2;
        else NXT_ST <= STATE1;
        end if;
      when STATE2 =>
        OUT1 <= "010";
        if IN1 = '1' then NXT_ST <= STATE3;
        else NXT_ST <= STATE2;
        end if;
      when others =>
        OUT1 <= "XXX"; NXT_ST <= STATE0;
    end case;
  end process;

end behave;
```

syn_use_carry_chain

This attribute is used to turn on or off the carry chain implementation for adders.

Verilog Syntax object synthesis syn_use_carry_chain = {1| 0} */ ;

Verilog Example To use this attribute globally, apply it to the module.

```
module test (a, b, clk, rst, d) /* synthesis
syn_use_carry_chain = 1 */;
```

VHDL Syntax attribute syn_use_carry_chain of object : objectType is true | false ;

VHDL Example

```
architecture archtest of test is
signal temp : std_logic;
signal temp1 : std_logic;
signal temp2 : std_logic;
signal temp3 : std_logic;
attribute syn_use_carry_chain : boolean;
attribute syn_use_carry_chain of archtest : architecture is
true;
```

syn_useenables

This attribute controls the use of clock enables on registers in the design. Usually exploiting clock enables on registers is beneficial. However, there are timing closure situations where clock enable routing causes timing violations. This is one reason why the user may want to stop the use of clock enable on the register.

Verilog Syntax: object /* synthesis syn_useenables = "0 | 1" */;

Verilog Example

```
module syn_useenable1 (Din1,Din2,en,clk,Dout);
  input [7:0] Din1, Din2;
  input clk,en;
  output [7:0] Dout;
  reg [7:0] temp1;
  reg [7:0] Dout /* synthesis syn_useenables = 0*/;
  always@(posedge clk)
    begin
      temp1 <= Din1 & Din2;
    end
  always @(posedge clk)
    begin
      if(en)
        Dout <= temp1;
    end
endmodule
```

VHDL Syntax: attribute syn_useenables of object : objectType is "true | false";

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;

entity syn_useenable2 is
  port(
     Din1 : in std_logic_vector(7 downto 0);
     Din2 : in std_logic_vector(7 downto 0);
     clk  : in std_logic;
     en   : in std_logic;
     Dout : out std_logic_vector(7 downto 0)
    );
end entity;

architecture behave of syn_useenable2 is
  signal temp1 : std_logic_vector(7 downto 0);
  attribute syn_useenables: boolean;
  attribute syn_useenables of Dout: signal is false;
begin
  process(clk)
  begin
    if clk'event and clk = '1' then
      temp1 <= Din1 and Din2;
    end if;
  end process;

  process(clk)
  begin
    if clk'event and clk = '1' then
      if en = '1' then
        Dout <= temp1;
      end if;
    end if;
  end process;
end behave;
```

syn_useioff

This attribute overrides the default behavior to pack registers into I/O pad cells based on timing requirements for the target Lattice families. Attribute syn_useioff is Boolean-valued: 1 enables (default) and 0 disables register packing. You can place this attribute on an individual register or port or apply it globally. When applied globally, the synthesis tool packs all input, output, and I/O registers into I/O pad cells. When applied to a register, the synthesis tool packs the register into the pad cell; and when applied to a port, it packs all registers attached to the port into the pad cell.

The syn_useioff attribute can be set on the following ports:

- top-level port
- register driving the top-level port
- lower-level port, if the register is specified as part of the port declaration

Verilog Syntax object /*synthesis syn_useioff = {1 | 0} */ ;

Verilog Example To use this attribute globally, apply it to the module.

```
module test (a, b, clk, rst, d) /* synthesis syn_useioff = 1 */;
```

To use this attribute on individual ports, apply it to individual port declarations.

```
module test (a, b, clk, rst, d);
input a;
input b /* synthesis syn_useioff = 1 */;
```

VHDL Syntax attribute syn_useioff of object : objectType is true | false ;

VHDL Example

```
architecture archtest of test is
signal temp : std_logic;
signal temp1 : std_logic;
signal temp2 : std_logic;
signal temp3 : std_logic;
attribute syn_useioff : boolean;
attribute syn_useioff of archtest : architecture is true;
```

translate_off/translate_on

This attribute allows you to synthesize designs originally written for use with other synthesis tools without needing to modify source code. All source code that is between these two attributes is ignored during synthesis.

Verilog Syntax /* pragma translate_off */
/* pragma translate_on */

Verilog Example

```
module real_time (ina, inb, out);
input ina, inb;
output out;
/* synthesis translate_off */
realtime cur_time;
/* synthesis translate_on */
assign out = ina & inb;
endmodule
```

VHDL Syntax pragma translate_off
pragma translate_on

VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
entity adder is
   port (a, b, cin:in std_logic;
         sum, cout:out std_logic );
end adder;
architecture behave of adder is
signal a1:std_logic;
--synthesis translate_off
constant a1:std_logic:='0';
--synthesis translate_on
begin
   sum <= (a xor b xor cin);
   cout <= (a and b) or (a and cin) or (b and cin); end behave;
```

Chapter 18

Lattice Module Reference Guide

IPexpress provides a variety of modules to assist your design work. These modules cover a variety of common functions and can be customized. They are optimized for Lattice device architectures. Use these modules to speed your design work and to get the most effective results.

PMI (Parameterized Module Instantiation) is an alternate way to use some of the modules that come with IPexpress. With PMI, instead of using IPexpress, you directly instantiate a module into your HDL and customize it by setting parameters in the HDL. You may find this easier than using IPexpress with the simpler modules.

This guide describes the modules that come with IPexpress and the related PMI modules. The descriptions mainly cover the options and controls of the configuration dialog boxes for the modules. When there is a related PMI module, the descriptions also include specifications for the PMI module's customizable parameters and ports.

See Also

- "Use PMI or IPexpress?" on page 191
- "Creating IPexpress Modules and IP" on page 191
- "Using PMI" on page 200

Finding Modules in This Guide

The module descriptions are arranged alphabetically. To find modules by their functional type, see the following lists. To look up a specific PMI module, see Table 144 on page 1402.

Architecture

I/O

- "DDR" on page 1417
- "DDR_GENERIC" on page 1419
- "DDR_MEM" on page 1428
- "DQS" on page 1447
- "GDDR_7:1" on page 1475
- "MIPI_DPHY" on page 1491
- "SDR" on page 1638

Other

- "Digital CDR" on page 1435
- "DLL" on page 1445
- "Dynamic Bank Controller" on page 1447
- "EFB" on page 1448
- "EXTREF" on page 1457
- "I2C" on page 1476
- "ORCAstra" on page 1555
- "PCS" on page 1556
- "PLL" on page 1573
- "PMU" on page 1600
- "Power Controller" on page 1600
- "Power Guard" on page 1601
- "System_Bus" on page 1659
- "Tag Memory" on page 1663
- "Mult_Add_Sub_Sum" on page 1504

Arithmetic

- "Adder" on page 1404
- "Adder_Subtractor" on page 1407
- "Comparator" on page 1411
- "Complex_Multiplier" on page 1412
- "Convert" on page 1414
- "Counter" on page 1415
- "FFT_Butterfly" on page 1457
- "LFSR" on page 1478
- "Mult_Add_Sub" on page 1500
- "Multiplier" on page 1544
- "Multiply_Accumulate" on page 1552
- "Sin-Cos_Table" on page 1643
- "Subtractor" on page 1656

DSP

- "1D_FILTER" on page 1403
- "ADDER_TREE" on page 1410
- "BARREL_SHIFTER" on page 1410
- "MAC" on page 1479
- "MMAC" on page 1492
- "MULT" on page 1492
- "MULTADDSUB" on page 1507
- "MULTADDSUBSUM" on page 1526
- "SLICE" on page 1644
- "WIDE_MUX" on page 1663

Memory

Distributed RAM

- "Distributed_DPRAM" on page 1436
- "Distributed_ROM" on page 1439
- "Distributed_SPRAM" on page 1442

EBR components

- "RAM_DP" on page 1605
- "RAM_DP_TRUE" on page 1614
- "RAM_DQ" on page 1625
- "ROM" on page 1634

Other

- "FIFO" on page 1458
- "FIFO_DC" on page 1462
- "RAM_Based_Shift_Register" on page 1602

Table 144: PMI Module Lookup

PMI Module Name	IPexpress Module Name
pmi_add	Adder
pmi_addsub	Adder_Subtractor
pmi_complex_mult	Multiplier
pmi_constant_mult	Multiplier
pmi_counter	Counter
pmi_distributed_dpram	Distributed_DPRAM
pmi_distributed_rom	Distributed_DPRAM
pmi_distributed_shift_reg	RAM_Based_Shift_Register
pmi_distributed_spram	Distributed_SPRAM
pmi_dsp_casmultaddsub	MULTADDSUB
pmi_dsp_mac	MAC
pmi_dsp_mult	MULT
pmi_dsp_multaddsub	MULTADDSUB
pmi_dsp_multaddsubsum	MULTADDSUBSUM
pmi_dsp_preadd_slice	SLICE
pmi_fifo	FIFO
pmi_fifo_dc	FIFO_DC
pmi_mac	Multiply_Accumulate
pmi_mult	Multiplier
pmi_multaddsub	Mult_Add_Sub
pmi_multaddsubsum	Mult_Add_Sub_Sum
Spmi_pll	PLL
pmi_pll_fp	PLL
pmi_ram_dp	RAM_DP
pmi_ram_dp_be	RAM_DP
pmi_ram_dp_true	RAM_DP_TRUE
pmi_ram_dp_true_be	RAM_DP_TRUE
pmi_ram_dq	RAM_DQ
pmi_ram_dq_be	RAM_DQ
pmi_rom	ROM
pmi_sub	Subtractor

1D_FILTER

1D FIR filter that can be either symmetrical or asymmetrical.

Table 145: 1D_FILTER Dialog Box

Feature	Description
Mode	Choose either symmetrical, asymmetrical serial, or asymmetrical parallel mode.
Number of Taps	Enter the number of taps wanted.
Symmetry	If in Symmetry mode, specify either positive or negative symmetry.
Reset Mode	Specify either a synchronous or asynchronous reset.
Dual Channel Mode	Select to double the DataA input and the Result output signals. DataA and DataB are automatically signed.
DATAA Width	Choose the width of the DataA input signal. The actual width depends on this value, the number of taps, and the mode: ▶ Asymmetry Serial: Width ▶ Asymmetry Parallel: Width x taps ▶ Symmetry: Width
DATAB Width	Choose the width of the DataB input signal. The actual width depends on this value, the number of taps, and the mode: ▶ Asymmetry: Width x taps ▶ Symmetry: Width x taps/2 (taps/2 is rounded up to an integer.)
DATAA, DATAB Sign	Specify whether the data input signals are signed or unsigned. Only available without high speed operation.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also ▶"Generating a Module or IP with IPexpress" on page 195

Adder

A two-input adder that performs signed or unsigned addition of the data from input ports DataA and DataB. If the optional port Cin is used, the addition result DataA[size-1..0] + DataB[size-1..0] + Cin is returned to the output port Result. For a description of some of the possible ports, see Table 147.

Table 146: Adder Dialog Box

Feature	Description
Specify the Data Width of the Adder	Specifies the input bus width of the module.
Specify the Representation of the Adder	Selections can be either signed or unsigned. Choosing the signed representation will keep the signed bit on the output. Unsigned will remove the signed bit from the output.
Complex Inputs	Allows implementation of complex adder.
Use Carry-in port	If selected will add a carry-in port to the module.
Specify the Carry-out Port	Use the carry-out port, overflow port, or none. Carry-out is available with unsigned addition. Overflow is available with signed addition.
Enable Output Register	Allows the implementation of registers on the outputs.
Specify number of pipeline stages	Specifies the number of register stages inserted into the adder. More stages improves achievable frequency of operation, but the latency for the output increases. To enable this option, the data width must be at least 9 bits
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 147: Adder Ports

If the Complex Inputs option is selected, each of these ports will come in pairs: *_Re (real part) and *_Im (imaginary part).

Name	Description
Cin	Carry-in to the low-order bit. (Optional)
DataA[size-1..0]	Augend.
DataB[size-1..0]	Addend
Result[size-1..0]	DataA[size-1..0] + DataB[size-1..0] + Cin

Table 147: Adder Ports (Continued)

If the Complex Inputs option is selected, each of these ports will come in pairs: *_Re (real part) and *_Im (imaginary part).

Name	Description
Cout	Carry-out of the most significant bit to detect overflow in unsigned representation. When Cout is prepended to Result, the result is a vector that always has sufficient precision to represent the result of the operation, {Cout, Result} = DataA + DataB + Cin. (Optional)
Overflow	Result exceeds available precision when used with signed representation. XOR of the carry into the most significant bit and Cout. (Optional)

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195

pmi_add

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_add.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA	Bus	(pmi_data_width - 1):0
I	DataB	Bus	(pmi_data_width - 1):0
I	Cin	Bit	N/A
O	Result	Bus	(pmi_result_width - 1):0
O	Cout	Bit	N/A
O	Overflow	Bit	N/A

Verilog pmi_add Definition

```
module pmi_add
  #(parameter pmi_data_width = 8,
    parameter pmi_result_width = 8,
    parameter pmi_sign = "off",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_add"
    )

    (
     input [pmi_data_width-1:0] DataA,
     input [pmi_data_width-1:0] DataB,
     input Cin,
```

```
     output [pmi_data_width-1:0] Result,
     output Cout,
     output Overflow)/*synthesis syn_black_box */;
endmodule // pmi_add
```

VHDL pmi_add Definition

```
component pmi_add is
     generic (
       pmi_data_width : integer := 8;
       pmi_result_width : integer := 8;
       pmi_sign : string := "off";
       pmi_family : string := "EC";
       module_type : string := "pmi_add"

    );
    port (
     DataA : in std_logic_vector(pmi_data_width-1 downto 0);
     DataB : in std_logic_vector(pmi_data_width-1 downto 0);
     Cin: in std_logic;
     Result : out std_logic_vector(pmi_data_width-1 downto 0);
     Cout: out std_logic;
     Overflow: out std_logic
   );
  end component pmi_add;
```

Parameter Names and Values

The following table contains the pmi_add module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_data_width	1 to 64	8	The LatticeSC/M devices' data and result widths range from 1 to 256.
pmi_result_width	1 to 64	8	The LatticeSC/M devices' data and result widths range from 1 to 256.
pmi_sign	"on" | "off"	"off"	
pmi_family	"EC" | "ECP" | "ECP2" | "ECP2M" | "ECP3" | "ECP5U" | "ECP5UM" | "SC" | "SCM" | "XP" | "XP2" | "XO" | "XO2" | "XO3L" | "LPTM"	"EC"	

See Also

- "Use PMI or IPexpress?" on page 191
- "Using PMI" on page 200

Adder_Subtractor

A two-input adder/subtractor that performs signed or unsigned addition/ subtraction of the data from input ports DataA and DataB based on the specification from the input port Add_Sub. If the optional port Cin is used, the addition/subtraction result DataA + DataB + Cin / DataA - DataB + Cin - 1 are returned to the output port Result. An optional Overflow port is provided to detect overflow conditions when doing operations on signed data. or a description of some of the possible ports, see Table 149 on page 1407.

Table 148: Adder_Subtractor Dialog Box

Feature	Description
Specify the Data Width of the Adder_Subtractor	Specifies the input bus width of the module.
Specify the Representation of the Adder_Subtractor	Selections can be either signed or unsigned. Choosing the signed representation will keep the signed bit on the output. Unsigned will remove the signed bit from the output.
Complex Inputs	Allows implementation of complex adder_subtractor.
Use Carry-in port	If selected will add a carry-in port to the module. This port will be Active High in Adder mode. This port will be Active Low in Subtractor mode.
Specify the Carry-out Port	Use the carry-out port, overflow port, or none at all. Overflow port is allowed only when the representation is signed. This port will be Active High in Adder mode. This port will be Active Low in Subtractor mode.
Enable Output Register	Allows the implementation of registers on the outputs.
Specify number of pipeline stages	Specifies the number of register stages inserted into the adder/subtractor. More stages improves achievable frequency of operation, but the latency for the output increases. To enable this option, the data width must be at least 9 bits.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 149: Adder_Subtractor Ports

Name	Description
Add_Sub	Add_Sub=1, then Result=DataA+DataB+Cin. Add_Sub=0, then Result=DataA-DataB+Cin-1.
Cin (optional)	Carry-in to the low-order bit.

Table 149: Adder_Subtractor Ports

Name	Description
DataA[size-1..0]	Augend (Add_Sub=1) / Minuend (Add_Sub=0)
DataB[size-1..0]	Addend (Add_Sub=1) / Subtrahend (Add_Sub=0)
Result[size-1..0]	Result (Add_Sub=1) = DataA + DataB + Cin Result (Add_Sub=0) = DataA - DataB + Cin - 1
Cout (optional)	Carry-out of the MSB to detect overflow in unsigned representation. When Cout is prepended to the Result, the result is a vector that always has sufficient precision to represent the result of the operation. {Cout, Result} = DataA + DataB + Cin (Add_Sub=1) {Cout, Result} = DataA - DataB + Cin - 1 (Add_Sub=0)
Overflow (optional)	Result exceeds available precision when used with signed representation. XOR of the carry into the MSB and Cout.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195

pmi_addsub

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_addsub.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA	Bus	(pmi_data_width - 1):0
I	DataB	Bus	(pmi_data_width - 1):0
I	Cin	Bit	N/A
I	Add_Sub	Bit	N/A
O	Result	Bus	(pmi_result_width - 1):0
O	Cout	Bit	N/A
O	Overflow	Bit	N/A

Verilog pmi_addsub Definition

```
module pmi_addsub #(parameter pmi_data_width = 8,
        parameter pmi_result_width = 8,
        parameter pmi_sign = "off",
        parameter pmi_family = "EC",
        parameter module_type = "pmi_addsub"
        )
```

```
  (
   input [pmi_data_width-1:0] DataA,
   input [pmi_data_width-1:0] DataB,
   input Cin,
   input Add_Sub,
   output [pmi_data_width-1:0] Result,
   output Cout,
   output Overflow)/*synthesis syn_black_box */;
endmodule // pmi_addsub
```

VHDL pmi_addsub Definition

```
component pmi_addsub is
     generic (
       pmi_data_width : integer := 8;
       pmi_result_width : integer := 8;
       pmi_sign : string := "off";
       pmi_family : string := "EC";
       module_type : string := "pmi_addsub"

    );
    port (
     DataA : in std_logic_vector(pmi_data_width-1 downto 0);
     DataB : in std_logic_vector(pmi_data_width-1 downto 0);
     Cin: in std_logic;
     Add_Sub: in std_logic;
     Result : out std_logic_vector(pmi_data_width-1 downto 0);
     Cout: out std_logic;
     Overflow: out std_logic
   );
  end component pmi_addsub;
```

Parameter Names and Values

The following table contains the pmi_addsub module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value
pmi_data_width	1 to 64	8
pmi_result_width	1 to 64	8
pmi_sign	“on”	“off”	“off”
pmi_family	“EC”	“ECP”	“ECP2”	“ECP2M”	“ECP3”	“ECP5U”	“ECP5UM”	“XP”	“XP2”	“XO”	“XO2”	“XO3L”	“LPTM”	“EC”

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

ADDER_TREE

Adder that can accept several input values simultaneously. The result is the same width as the inputs, so you must control overflow.

Table 150: ADDER_TREE Dialog Box

Feature	Description
Width	Specifies the width of the input data buses and of the Result bus.
Number of Inputs	Specifies the number of input data buses.
Reset Mode	Specify either a synchronous or asynchronous reset.
Enable Fully Pipelined Mode	Enables the output registers of all stages.
Enable Input Register	Allows the implementation of registers on the inputs.
Enable Output Register	Allows the implementation of a register on the output.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195

BARREL_SHIFTER

Shifts a data word multiple positions in a single clock cycle.

Table 151: BARREL_SHIFTER Dialog Box

Feature	Description
Shift Direction	Specify the direction to shift the data.
Type	Select the type of shift: Zero Insert, Sign Extension, or Rotate. Sign Extension is available only when the shift direction is to the right.
Data Width	Specify the width of the input and output data buses.
Maximum Number of Shifts	Specify the maximum number of bits that can be shifted in a single clock cycle. This value can be up to one less than the data width.
Reset Mode	Choose between synchronous and asynchronous reset.
Enable Input Register	Adds registers to the input bus.

Table 151: BARREL_SHIFTER Dialog Box (Continued)

Feature	Description
Enable Pipeline Register	Adds pipeline registers.
Enable Output Register	Adds registers to the output bus.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

▶ “Generating a Module or IP with IPexpress” on page 195

Comparator

A two-input comparator that performs signed or unsigned comparison of the value represented by DataA versus the value represented by DataB, the electrical equivalent of bit-pattern comparison operation. The module outputs the results of comparisons in terms of larger than, smaller than, equal to, or combinations of these operations.

Table 152: Comparator Dialog Box

Feature	Description
Specify the data width of the comparator	Specifies the width of the comparator.
Specify the representation of comparator	Specify the input representation to be signed or unsigned. For signed inputs, the output depends on the sign bit.
Specify the output port compare function	Specify the output port compare function. This function can be A != B, A < B, A <= B, A = B, A > B, A >= B.
Use LUT based implementation	Uses a LUT-based implementation to use fewer resources. Available only with the A = B and A != B functions.
Enable Output Register	Allows the implementation of registers on the outputs.
Specify number of pipeline stages	Specifies the number of register stages inserted into the comparator. More stages improves achievable frequency of operation, but the latency for the output increases. To enable this option, the data width must be at least 9 bits.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.

Table 152: Comparator Dialog Box (Continued)

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195

Complex_Multiplier

Takes two inputs and multiplies them. For example, y1 = a1 + jb1 and y2 = a2 + jb2 - the product is a1a2 - b1b2 + j(a1b2 + a2b1).

Look-up table (LUT) implementation is available. For DSP devices, DSP implementation is also available.

Table 153: Complex_Multiplier Dialog Box

Feature	Description
Block Implementation	Allows you to choose between LUT-based implementation and DSP. DSP is only available with ECP5, LatticeECP, LatticeECP2/M, LatticeECP3, and LatticeXP2.
Input A Width	Bit size of Multiplier input. Bit width range is 2 to 36. The default value is 8.
Input B Width	Bit size of Multiplicand input. Bit width range is 2 to 36. The default value is 8.
Product Width	Bit size of product, equal to Input A Width + Input B Width + 1. This field is read-only.
Representation	Selections can be either signed or unsigned. Choosing the signed representation will keep the signed bit on the output. Unsigned will remove the signed bit from the output.
Specify the Number of Pipeline Stages	Specifies the number of register stages inserted into the complex multiplier. The default value is 1.
Enable Input Register	Allows you to register the data inputs. Default is enabled.
Enable Output Registers	Allows you to register the data outputs. Default is enabled.
Implementation	**3 Multiplier** - Implements the complex multiplier using three multipliers but requires more implementation logic. **4 Multiplier** - Implements the complex multiplier using four multipliers.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.

Table 153: Complex_Multiplier Dialog Box (Continued)

Feature	Description
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195

Convert

Converts from one format to another, performing rounding or saturation as required.

Table 154: Convert Dialog Box

Feature	Description
Input	**Width** - Number of bits in the input word. A positive integer that specifies the total number of bits that will be used to represent the data in twos complement format. Valid values are from 1 to 256. **Binary Point** - A positive integer that specifies the position of the binary point in the input data word. Valid values for unsigned word are from 0 to the output width. Valid values for signed word are from 0 to the output width minus 1. The MSB is treated as the sign bit. **Sign** - The format of the data value. The available values are signed and unsigned.
Output	**Width** - Number of bits in the output word. A positive integer that specifies the total number of bits that will be used to represent the data in twos complement format. Valid values are from 1 to 256. **Binary Point** - A positive integer that specifies the position of the binary point in the output data word. Valid values for unsigned word are from 0 to the output width. Valid values for signed word are from 0 to the output width minus 1. The MSB is treated as the sign bit. **Sign** - The format of the data value. The available values are signed and unsigned. **Rounding** - Three settings exist and determine the manner in which the input format value is shortened to the output format: Truncate, Nearest, and Convergent. - **Truncate** shortens the format by simply discarding the excess bits of the larger input format over the output format. - **Nearest** adds or subtracts an LSB if the excess bits sum to an absolute value that exceeds an LSB of the output format. - **Convergent** is the same as Nearest except when the excess bits add to exactly one-half an LSB, in which case a refinement to Nearest algorithm is employed. If the number of output fractional bits is less than the input, then quantization will be done according to this selection. Note that choosing any selection other than Truncate will result in an increase in the hardware requirements and potentially a lower overall performance. **Saturate** - If the number of integer output bits is less than that of the input, then any overflow or underflow will be treated according to this selection. - **Wrap** discards excess bits. - **Min_Max** sets to minimum or maximum value.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified and saves the files in the project folder.
Close	Cancels the dialog box without generating the module.
Help	Opens the Help Contents.

See Also

▶ "Generating a Module or IP with IPexpress" on page 195

Counter

An up, up-down, or down counter. For a description of some of the possible ports, see Table 156 on page 1415.

Table 155: Counter Dialog Box

Feature	Description
Specify the data width of the counter	Specifies the input bus width of the module.
Specify the direction of the counter	Selects the counter direction to be Up, Down, or Up-Down. Up-Down is not available with LatticeSC/M.
Optimized for speed	Allows you to improve speed by using more logic resources. Only available with data widths of 33 bits or more. Not available with the Up-Down direction.
Lower count value	This specifies the lower count value of the counter. This is a double-precision number that will be converted to the specified format. If the value exceeds the range achievable by the selected format an error will be issued. Only available with data widths of 32 bits or fewer.
Upper count value	This specifies the upper count value of the counter. This is a double-precision number that will be converted to the specified format. If the value exceeds the range achievable by the selected format an error will be issued. For limited counting, this value must be an even multiple of the count range. Otherwise, an error will be issued. Only available with data widths of 32 bits or fewer.
Enable load input	Check this box to provide load and data input pins to the block. When the load pin goes high, the data on the data in pin will be loaded into the counter. If an enable pin is provided, it must also be high in order to load the data. With most devices, this option is only available with data widths of 32 bits or fewer. With MachXO and Platform Manager, this option is available with all data widths.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 156: Counter Ports

Name	Description
Clock	Positive, edge-triggered clock input
Clk_En	Clock Enable

Table 156: Counter Ports

Name	Description
Aclr	Asynchronous clear input
UpDown (Optional)	Controls the direction of the counter when **UpDown** direction option is selected: 1 for up and 0 for down.
Load (Optional)	Enables the loading of the counter from LData. Added when **Enable load input** check box is selected.
Ldata (Optional)	Parallel load value for counter. Added when **Enable load input** check box is selected.
Q[size-1..0]	Counter output

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195

pmi_counter

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_counter.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Clock	Bit	N/A
I	Clk_En	Bit	N/A
I	Aclr	Bit	N/A
I	UpDown	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_counter Definition

```
module pmi_counter
  #(parameter pmi_data_width = 8,
    parameter pmi_updown = "up",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_counter"
    )

  (
   input Clock,
   input Clk_En,
   input Aclr,
   input UpDown,
   output [pmi_data_width-1:0] Q)/*synthesis syn_black_box */;
endmodule // pmi_counter
```

VHDL pmi_counter Definition

```
component pmi_counter is
     generic (
       pmi_data_width : integer := 8;
       pmi_updown : string := "up";
       pmi_family : string := "EC";
       module_type : string := "pmi_counter"

    );
    port (
     Clock: in std_logic;
     Clk_En: in std_logic;
     Aclr: in std_logic;
     UpDown: in std_logic;
     Q : out std_logic_vector(pmi_data_width-1 downto 0)
   );
  end component pmi_counter;
```

Parameter Names and Values

The following table contains the pmi_counter module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_data_width	1 to 64	8	The LatticeECP/EC, LatticeECP2, LatticeXP, Platform Manager, and MachXO device families support a data width range from 1 to 64. The data width range for LatticeSC/M device families is from 1 to 256.
pmi_updown	“up” | “down” | “up-down”	“up”	Up-down mode is not supported for the LatticeSC/M devices.
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3H” | “LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

DDR

A DDR of specified width and mode. Although a DDR generated for a specific width can be used for other widths, it usually requires extra care in specifying the connection while instantiating.

The module dialog box has two tabs:

- Configuration, below

- Advanced, see Table 158 on page 1419

Table 157: DDR Dialog Box—Configuration Tab

Feature	Description
Mode	Specify the Mode of the DDR. The DDR can be used as an input, output, tristate, or bidirectional. A tristate mode consists of two output DDRs. A bidirectional mode has one input and two output DDRs. Values: Input, Output, Bidirectional, Tristate
Bus Width	Specify the width of the desired DDR. Values: 1 to 64
Gearing Ratio	Specify the gearing ratio. A 1x is the regular DDR where the output width is twice the input bus width. For gear ratio of 2x and 4x, the output width is 4 or 8 times the input width. The clock rate will be slower by the same amount.
Input Clock Mode	Specifies the type of input clock to use when no gearing is used. Values: Edge, Primary/Secondary
Output Clock Mode	Specifies the type of output clock to use when no gearing is used. Values: Edge, Primary/Secondary
Data Delay	Specify the Data Delay type. By default, there is no delay in the data path. The delay may be set at configuration to values specified by CDEL and FDEL, set to match the primary or edge clock, or set to match the DLL Delay Controls (DCNTL bus). Values: None, DLL, Edge Clock, Primary Clock, ECLK Injection, User Defined
CDEL	Specify the coarse delay setting. Values: 0, 1, 2, 3
FDEL	Specify the fine delay setting. Values: 0 to 47
Use CLKDIV	Specify the use of CLKDIV element to get the slower clock. Using the clock and reset from the CLKDIV will reduce the skew between the two. This is useful for high-speed interfaces that have gearing. This option is only available for Input DDR with a gearing ratio of 2x or 4x.
Use Single Clock for 1x	Specify that only one clock is required for the DDR module. This option is only available for Input and Bidirectional DDR with a gearing ratio of 1x.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 158: DDR Dialog Box—Advanced Tab

Feature	Description
Use AIL	Specify that the Adaptive Input Logic (AIL) is used. This is available for the Input and Input DDRs of bidirectional DDRs. The AIL is the ability of the I/O logic to automatically control coarse and fine delay settings to guarantee setup and hold times for a single I/O by sliding the data to a particular clock edge. The AIL features are not bus-based, but rather, per input port. Since the AIL will slide the data to a particular clock edge, the bits of a bus may not be latched on the same clock edge. For this reason, a training pattern will be required to realign the individual bits of the input data bus to the same clock edge. For single bit instances, this training pattern is not required. An AIL deskew reference design is available from Lattice Semiconductor that provides an example of the circuitry and training pattern required to align the bits of a bus. Please contact Lattice Semiconductor for more information on this reference design.
AIL Acquisition Window (ps)	Specify the window size in ps where data transitions are not allowed. The AIL continuously delays data to keep the AIL Acquisition Window within the data pulse width away from the transition edges. Values: 400, 720, 1040, 1360
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1099, *LatticeSC DDR/DDR2 SDRAM Memory Interface User's Guide*

DDR_GENERIC

Output DDR for a generic application, built with output DDR (ODDRXB) primitives to support high-speed DDR interface implementations.

The options vary with the device family. See:

- "DDR_GENERIC Options (ECP5)" on page 1420
- "DDR_GENERIC Options (LatticeEC, LatticeECP, LatticeXP)" on page 1422
- "DDR_GENERIC Options (LatticeECP2/M, LatticeXP2)" on page 1423
- "DDR_GENERIC Options (LatticeECP3)" on page 1424

- "DDR_GENERIC Options (MachXO2, MachXO3L, Platform Manager 2)" on page 1426

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1265, *ECP5 High-Speed I/O Interface*
- TN1050, *LatticeECP/EC and LatticeXP DDR Usage Guide*
- TN1105, *LatticeECP2/M High-Speed I/O Interface*
- TN1180, *LatticeECP3 High-Speed I/O Interface*
- TN1138, *LatticeXP2 High-Speed I/O Interface*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1281, *Implementing High-Speed Interfaces with MachXO3L Devices*
- TN1301, *CrossLink High-Speed I/O Interface*

DDR_GENERIC Options (ECP5)

The module dialog box has two tabs:

- Pre-Configuration, below
- Configuration, see Table 160 on page 1421

First, fill out the Pre-Configuration tab. The Configuration tab is automatically filled out based on the specifications in the Pre-Configuration tab. This is sufficient for most uses. Use the Configuration tab only if you want to further customize the module.

Table 159: DDR_GENERIC Dialog Box for ECP5—Pre-Configuration Tab

Feature	Description
Interface Type	Specify whether the interface is for transmit or receive.
Enable Tri-state Control	Available only with the Transmit interface type.
I/O Standard for this Interface	Choose an I/O standard.
Bus Width for this Interface	Specify the width of the bus in bits. The maximum possible bus width depends on the I/O standard.
Clock Frequency for this Interface	Specify the frequency of the clock in megahertz.
Interface Bandwidth	Shows the resulting bandwidth in megabits/second. Bandwidth is the bus width multiplied by the clock frequency times 2. This field is read-only.
Clock to Data Relationship at the Pins	Choose the clock-to-data relationship.

Table 159: DDR_GENERIC Dialog Box for ECP5—Pre-Configuration Tab (Continued)

Feature	Description
Next	Displays the Configuration tab.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 160: DDR_GENERIC Dialog Box for ECP5—Configuration Tab

Feature	Description
Interface selection based on pre-configuration	After filling out the Pre-Configuration tab, clear this option to further customize the module. Select this option to change the settings to match the Pre-Configuration tab.
Interface Type	Specify whether the interface is for transmit or receive.
Enable Tri-state Control	Available only with the Transmit interface type.
I/O Standard	Choose an I/O standard.
Clock Frequency	Specify the frequency of the clock in megahertz.
Gearing Ratio	Choose the gearing ratio. 4:1 is required for frequencies greater than 200 MHz. For a gearing ratio of 4:1, the output width is four times the input width.
Alignment	Choose the clock-to-data relationship.
Bus Width	Specify the width of the bus in bits. The maximum possible bus width depends on the I/O standard.
Organize Parallel Data	Specify how to organize parallel data.
Enable ECLK Bridge	If the bus is going to cross sides of the device, select this option to include an ECLK bridge.
Interface	Shows the interface in terms of primitives. This field is read-only.
Data Path Delay	Shows the delay type: ▶ Bypass means no delay. (Transmit only.) ▶ Static sets a fixed delay. ▶ Dynamic adds a data_direction, data_loadn, data_move, and data_cflag ports each of which is as wide as the Bus Width. (Transmit only.) ▶ User Defined allows you to set a delay value in the Delay Value for User Defined field. Not available with the Receive MIPI interface type.
Delay Value for User Defined	If Data Path Delay is “User Defined,” specify the number of fine delay increments. Refer to the device datasheet to see the delay value for each increment.
Enable Dynamic Margin Control on Clock Delay	Select to enable dynamic margin control on clock delay.

Table 160: DDR_GENERIC Dialog Box for ECP5—Configuration Tab (Continued)

Feature	Description
Generate PLL with this Module	Select to include a PLL (Phase Lock Loop). Available only with the Transmit interface type. Specify the **PLL Input Clock Frequency** in megahertz and click **Calculate**. Actual Clock Frequency shows the result.
PLL Reference Clock From I/O Pin	Select if the PLL reference clock will come from an I/O pin. Then specify the CLKI Input Buffer Type (below). Only available when Generate PLL with this Module (above) is selected.
CLKI Input Buffer Type	Choose the buffer for CLKI, the input reference clock for the PLL, when the clock is coming from an I/O pin.
Reference Clock From I/O Pin	Select if the reference clock will come from an I/O pin. Then specify the Reference Clock Input Buffer Type (below). Available only with the Transmit interface type.
Reference Clock Input Buffer Type	Choose the buffer for the input reference clock when the clock is coming from an I/O pin.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1265, *ECP5 High-Speed I/O Interface*

DDR_GENERIC Options (LatticeEC, LatticeECP, LatticeXP)

Table 161: DDR_GENERIC Dialog Box for LatticeEC, LatticeECP, and LatticeXP

Feature	Description
DDR Data Width	Specify the width of the DDR bus. Values: 1 to 64
Tristate Enable for DDR Data Output Registers	Specify if the DDR output is a tristate bus. In this case, a DDR tristate enable, implemented using output DDRs, is generated. Otherwise, it is treated as an output bus.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.

Table 161: DDR_GENERIC Dialog Box for LatticeEC, LatticeECP, and LatticeXP (Continued)

Feature	Description
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1050, *LatticeECP/EC and LatticeXP DDR Usage Guide*

DDR_GENERIC Options (LatticeECP2/M, LatticeXP2)

Table 162: DDR_GENERIC Dialog Box for LatticeECP2/M, and LatticeXP2

Feature	Description
Mode	Specify the Mode of the DDR. The DDR can be used as an input, output, tristate, or bidirectional. A tristate mode consists of two output DDRs. A bidirectional mode has one input and two output DDRs. Values: Input, Output, Bidirectional, Tristate
DDR Data Width	Specify the width of the DDR bus. Values: 1 to 64
Gearing Ratio	Specify the gearing ratio. A 1x is the regular DDR where the output width is twice the input bus width. For gear ratio of 2x, the output width is 4 times the input width. The clock rate will be slower by the same amount. Values: 1x, 2x
Delay	Specify the Data Delay type. By default, Dynamic delay control is enabled. Values: Dynamic (default), Fixed XGMII, User Defined Note that availability of Delay control options are dependent upon Gearing Ratio selection. In 1x mode, you can select a fixed delay for the XGMII interface, a user-defined value between 0 and 15, or dynamic control using del[3:0]. In 2x mode, only the user-defined and dynamic control selections are available.
Multiplier for Fixed Delay	Specifies the delay in increments of approximately 50 ps. Values: 0 to 15
Use Single Clock for 1x	Specify that only one clock is required for the DDR module. This option is only available for Input and Bidirectional DDR with a gearing ratio of 1x.
Use CLKDIV for 2x	Specify the use of CLKDIV element to get the slower clock. Using the clock and reset from the CLKDIV will reduce the skew between the two. This is useful for high-speed interfaces that have gearing. This option is only available for Input DDR with a gearing ratio of 2x.

Table 162: DDR_GENERIC Dialog Box for LatticeECP2/M, and LatticeXP2 (Continued)

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1105, *LatticeECP2/M High-Speed I/O Interface*
- TN1138, *LatticeXP2 High-Speed I/O Interface*

DDR_GENERIC Options (LatticeECP3)

The module dialog box has two tabs:

- Pre-Configuration, below
- Configuration, see Table 164 on page 1425

First, fill out the Pre-Configuration tab. The Configuration tab is automatically filled out based on the specifications in the Pre-Configuration tab. This is sufficient for most uses. Use the Configuration tab only if you want to further customize the module.

Table 163: DDR_GENERIC Dialog Box for LatticeECP3—Pre-Configuration Tab

Feature	Description
Interface Type	Specify whether the interface is for transmit or receive.
Enable Tri-state Control	Available only with the Transmit interface type.
I/O Standard for this Interface	Choose an I/O standard.
Number of these interfaces on a side of a device	You can specify more than one interface in the module. All of the interfaces will be placed on the same side of the FPGA.
Bus Width for this Interface	Specify the width of the bus in bits. The maximum possible bus width depends on the I/O standard and the number of interfaces specified above.
Clock Frequency for this Interface	Specify the frequency of the clock in megahertz.
Interface Bandwidth (Calculated)	Shows the resulting bandwidth in megabits/second. Bandwidth is the bus width multiplied by the clock frequency times 2. This field is read-only.

Table 163: DDR_GENERIC Dialog Box for LatticeECP3—Pre-Configuration Tab (Continued)

Feature	Description
Clock to Data Relationship at the Pins	Choose the clock-to-data relationship.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 164: DDR_GENERIC Dialog Box for LatticeECP3—Configuration Tab

Feature	Description
Interface selection based on pre-configuration	After filling out the Pre-Configuration tab, clear this option to further customize the module. Select this option to change the settings to match the Pre-Configuration tab.
Interface Type	Specify whether the interface is for transmit or receive.
Enable Tri-state Control	Available only with the Transmit interface type.
I/O Standard	Choose an I/O standard.
Clock Frequency	Specify the frequency of the clock in megahertz.
Gearing Ratio	Choose the gearing ratio. 2x is required for frequencies greater than 200 MHz. For a gearing ratio of 2x, the output width is four times the input width.
Alignment	Choose the clock-to-data relationship.
Number of interfaces	You can specify more than one interface in the module. All of the interfaces will be placed on the same side of the FPGA.
Bus Width	Specify the width of the bus in bits. The maximum possible bus width depends on the I/O standard and the number of interfaces specified above.
Phase Adjust	Choose whether to use a PLL or DLL to adjust the clock phase. Available only with a Receive interface, 1x gearing ratio, and edge-to-edge alignment. See “DELAYB” on page 1890 and “TRDLLB” on page 2426.
Clock Divider	Choose the FPGA library primitive to use as the clock divider (SCLK). Available only with a Receive interface, 2x gearing ratio, and edge-to-edge alignment. See “CLKDIVB” on page 1870 and “TRDLLB” on page 2426.
Input Clock Multiplier	Available only with a Transmit interface and with a 2x gearing ratio.
PLL Input Clock Frequency	Shows the frequency resulting from the Clock Frequency and the Input Clock Multiplier. Available only with a Transmit interface, a 2x gearing ratio, and PLL selected in the Phase Adjust box. Read only.
Interface	Shows the interface in terms of primitives. This field is read-only.

Table 164: DDR_GENERIC Dialog Box for LatticeECP3—Configuration Tab (Continued)

Feature	Description
Data Path Delay	Shows the delay type: ▶ Bypass means no delay. ▶ Dynamic adds a 4-bit control port, del[3:0]. The value on the control port specifies the number of fine delay increments. Each increment is 1/16 of the clock cycle. ▶ Fixed sets a fixed delay. Available only with the receive interface type. This field is read-only.
DQS Grouping	**Number of DQS** - Shows the number of DQS groups, which is equal to the number of interfaces in the module. This field is read-only. **DQS Group** - Choose the number of bits in the group to use.
ISI Calibration	Choose the ISI calibration type.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1180, *LatticeECP3 High-Speed I/O Interface*

DDR_GENERIC Options (MachXO2, MachXO3L, Platform Manager 2)

The module dialog box has two tabs:

- Pre-Configuration, below
- Configuration, see Table 166 on page 1427

First, fill out the Pre-Configuration tab. The Configuration tab is automatically filled out based on the specifications in the Pre-Configuration tab. This is sufficient for most uses. Use the Configuration tab only if you want to further customize the module.

Table 165: DDR_GENERIC Dialog Box for MachXO2, MachXO3L, and Platform Manager 2—Pre-Configuration Tab

Feature	Description
Interface Type	Specify whether the interface is for transmit or receive and regular or MIPI (Mobile Industry Processor Interface).
Enable Tri-state Control	Available only with the Transmit interface type.
Enable MIPI High Speed Mode Only	Available only with the MIPI interface types.
I/O Standard for this Interface	Choose an I/O standard. Not available with MIPI interface types.
Clock Frequency for this Interface	Specify the frequency of the clock in megahertz.
Bus Width for this Interface	Specify the width of the bus in bits. The maximum possible bus width depends on the I/O standard and the number of interfaces specified above.
Number of this interface	This read-only field is calculated based on the selected Interface Type, Clock Frequency for this Interface, and Bus Width for this Interface. All of the interfaces will be placed on the same side of the FPGA.
Interface Bandwidth	Shows the resulting bandwidth in megabits/second. Bandwidth is the bus width multiplied by the clock frequency times 2. This field is read-only.
Clock to Data Relationship at the Pins	Choose the clock-to-data relationship.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 166: DDR_GENERIC Dialog Box for MachXO2, MachXO3L, and Platform Manager 2—Configuration Tab

Feature	Description
Interface selection based on pre-configuration	After filling out the Pre-Configuration tab, clear this option to further customize the module. Select this option to change the settings to match the Pre-Configuration tab.
Interface Type	Specify whether the interface is for transmit or receive.
Enable Tri-state Control	Available only with the Transmit interface type.
Enable MIPI High Speed Mode Only	Available only with the MIPI interface types.
I/O Standard	Choose an I/O standard. Not available with MIPI interface types.
Clock Frequency	Specify the frequency of the clock in megahertz.
Gearing Ratio	Choose the gearing ratio. 2x is required for frequencies greater than 200 MHz. For a gearing ratio of 2x, the output width is four times the input width.

Table 166: DDR_GENERIC Dialog Box for MachXO2, MachXO3L, and Platform Manager 2—Configuration Tab (Continued)

Feature	Description
Alignment	Choose the clock-to-data relationship.
Bus Width	Specify the width of the bus in bits. The maximum possible bus width depends on the I/O standard and the number of interfaces specified above.
Number of interfaces	You can specify more than one interface in the module. All of the interfaces will be placed on the same side of the FPGA.
Interface	Shows the interface in terms of primitives. This field is read-only.
Data Path Delay	Shows the delay type: ▶ Bypass means no delay. ▶ Predefined sets a fixed delay. ▶ User defined allows you to select a delay value from the next menu. ▶ Dynamic adds a 4-bit control port, del[3:0]. The value on the control port specifies the number of fine delay increments. Each increment is 1/16 of the clock cycle. Available only with the receive interface type.
Generate PLL with this Module	Select to generate a PLL as part of the module. Only available with Transmit and Transmit_MIPI modes.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Use Dedicated DDR Memory Support” on page 1165
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1281, *Implementing High-Speed Interfaces with MachXO3L Devices*

DDR_MEM

Interface to a DDR SDRAM memory. The module includes the bidirectional port and the associated clocking scheme.

The options vary with the device family. See:

- “DDR_MEM Options (ECP5)” on page 1429
- “DDR_MEM Options (LatticeEC, LatticeECP, LatticeXP)” on page 1431
- “DDR_MEM Options (LatticeECP2/M, LatticeXP2)” on page 1432

- "DDR_MEM Options (LatticeECP3, MachXO2, Platform Manager 2)" on page 1433

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1265, *ECP5 High-Speed I/O Interface*
- TN1050, *LatticeECP/EC and LatticeXP DDR Usage Guide*
- TN1105, *LatticeECP2/M High-Speed I/O Interface*
- TN1180, *LatticeECP3 High-Speed I/O Interface*
- TN1138, *LatticeXP2 High-Speed I/O Interface*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1301, *CrossLink High-Speed I/O Interface*

DDR_MEM Options (ECP5)

The module dialog box has three tabs:

- Configuration, below
- Clock/Address/Command, see Table 168 on page 1430
- Advanced Settings, for DDR2 only, Table 169 on page 1430

Table 167: DDR_MEM Dialog Box for ECP5—Configuration Tab

Feature	Description
Interface	Specify the mode of the DDR memory interface. This selection affects the I/O Buffer configuration, the frequency of DQS, and signaling of DQS.
I/O Buffer Configuration	Specify the I/O buffer type for the DDR memory interface.
DDR Memory Frequency	Select the frequency of operation of the DDR memory interface. The available values depend on the Interface type.
System Clock Frequency	Shows the required system clock frequency.
DQS Buffer Configuration for DDR2	If the interface is DDR2, specify whether the buffer is single-ended or differential.
Number of DQ per DQS	Select the number of DQ per DQS group.
Data Width	Specify the combined width of the DQS groups.
Total number of DQS Groups	Choose the number DQS groups based on the data width and the number of DQ.
Data Mask	Select this option to add a data mask port for each DQS group.
Clock/Address/Command	Select this option to fill in the Clock/Address/Command tab to further customize the module.

Table 167: DDR_MEM Dialog Box for ECP5—Configuration Tab (Continued)

Feature	Description
Enable Dynamic Margin Control on Clock Delay	Select to enable dynamic margin control on clock delay.
Generate PLL with this module	Select to include a PLL (Phase Lock Loop). Available only with the Transmit interface type. Specify the **PLL Input Clock Frequency** in megahertz and click **Calculate**. Actual Clock Frequency shows the result.
PLL Reference Clock From I/O Pin	Select if the PLL reference clock will come from an I/O pin. Then specify the CLKI Input Buffer Type (below). Only available when Generate PLL with this Module (above) is selected.
CLKI Input Buffer Type	Choose the buffer for CLKI, the input reference clock for the PLL.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 168: DDR_MEM Dialog Box for ECP5—Clock/Address/Command Tab

Feature	Description
Number of Clocks	Choose the number of clocks.
Address Width	Choose the width of the address in bits.
Bank Address Width	Choose the width of the bank address in bits.
Number of Chip Selects	Choose the number of bits in the chip select port.
Number of ODT	Choose the number of bits in the input and output on-die termination (ODT) ports.
Number of Clock Enables	Choose the number of bits in the input and output clock enable ports.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 169: DDR_MEM Dialog Box for ECP5—Advanced Settings Tab

Feature	Description
DQS Read Delay Adjustment DQS Write Delay Adjustment	Select whether the delay is PLUS or MINUS. Then select a delay value.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.

Table 169: DDR_MEM Dialog Box for ECP5—Advanced Settings Tab (Continued)

Feature	Description
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1265, *ECP5 High-Speed I/O Interface*

DDR_MEM Options (LatticeEC, LatticeECP, LatticeXP)

Table 170: DDR_MEM Dialog Box for LatticeEC, LatticeECP, and LatticeXP

Feature	Description
I/O Buffer Configuration	Specify the I/O buffer type for the DDR Memory interface. Values: SSTL25_I, SSTL25_II
DDR Data Width	Specify the width of the DDR bus. Values: 8, 16, 32, 64
Number of DQS	Specifies the number of DQS signals. There will be one DQS for every byte (8 bits) of data. Values: 1, 2, 4, 8
Frequency of DQS (MHz)	Select the frequency of operation of the DDR memory interface. Values: 100, 133, 166, 200
Lock/Jitter Sensitivity	Specify the Lock/Jitter Sensitivity for the Master DLL. Values: HIGH, LOW
LSR for DDR Input Registers	Specify if the input registers are of Reset or Set type. Values: SET, RESET
Create Clock Enable for DDR Data Input Registers	Specify if the clock enable port should be generated for the input DDR registers.
Tristate Enable for DDR Data Output Registers	Specify if the tristate enable signal of the output DDR (part of the bidirectional bus) data bus is DDR. Otherwise, this control signal is a single bit.
DDR Tristate Enable for DQS	Specify if the tristate enable signal of the DQS buffer is a DDR signal or a single bit.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.

Table 170: DDR_MEM Dialog Box for LatticeEC, LatticeECP, and LatticeXP (Continued)

Feature	Description
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1050, *LatticeECP/EC and LatticeXP DDR Usage Guide*

DDR_MEM Options (LatticeECP2/M, LatticeXP2)

Table 171: DDR_MEM Dialog Box for LatticeECP2/M, and LatticeXP2

Feature	Description
Mode	Specify the mode of the DDR Memory interface. Specify if it is a DDR or DDR2 interface. This selection affects the I/O Buffer configuration, the frequency of DQS, and signaling of DQS.
I/O Buffer Configuration	Specify the I/O buffer type for the DDR Memory interface. Values: SSTL25_I, SSTL25_II (for DDR) Values: SSTL18_I, SSTL18_II (for DDR2)
DDR Data Width	Specify the width of the DDR bus. Values: 8, 16, 18, 32, 64
Number of DQS	Specifies the number of DQS signals. There will be one DQS for every byte (8 bits) of data. Values: 1, 2, 4, 8
Frequency of DQS (MHz)	Select the frequency of operation of the DDR memory interface. Values: 100, 133, 166, 200 (for DDR) Values: 166, 200, 266 (for DDR2)
Lock/Jitter Sensitivity	Specify the Lock/Jitter Sensitivity for the Master DLL. Values: HIGH, LOW
LSR for DDR Input Registers	Specify if the input registers are of Reset or Set type. Values: SET, RESET
DDR Tristate Enable for DDR Data Output Registers	Specify if the tristate enable signal of the output DDR (part of the bidirectional bus) data bus is DDR. Otherwise, this control signal is a single bit.
DDR Tristate Enable for DQS	Specify if the tristate enable signal of the DQS buffer is a DDR signal or a single bit.

Table 171: DDR_MEM Dialog Box for LatticeECP2/M, and LatticeXP2 (Continued)

Feature	Description
DQS Buffer Configuration for DDR2	This option allows you to select either Single-ended or Differential. If Differential is selected, SSTL18D_II will be applied. By default, this option is grayed-out. If DDR2 is selected for the mode, then the I/O Buffer Configuration drop down will show SSTL18_I and SSTL18_II. If the selected I/O buffer configuration is SSTL18_II then this option will be active.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165
- TN1105, *LatticeECP2/M High-Speed I/O Interface*
- TN1138, *LatticeXP2 High-Speed I/O Interface*

DDR_MEM Options (LatticeECP3, MachXO2, Platform Manager 2)

The module dialog box has two tabs:

- Configuration, below
- Clock/Address/Command, see Table 173 on page 1434

Table 172: DDR_MEM Dialog Box for LatticeECP3, MachXO2, and Platform Manager 2—Configuration Tab

Feature	Description
Interface	Specify the mode of the DDR Memory interface. This selection affects the I/O Buffer configuration, the frequency of DQS, and signaling of DQS.
I/O Buffer Configuration	Specify the I/O buffer type for the DDR Memory interface.
DDR Memory Frequency	Select the frequency of operation of the DDR memory interface.
Number of DQS	Specify the number of DQS signals. There should be one DQS for every byte (8 bits) of data.
DQS Group	Choose the number of bits in the group to use.
DQS Buffer Configuration	If the interface is DDR2, specify whether the buffer is single-ended or differential.

Table 172: DDR_MEM Dialog Box for LatticeECP3, MachXO2, and Platform Manager 2—Configuration Tab (Continued)

Feature	Description
Data Width	Shows the combined width of the DQS groups. This field is read-only. (MachXO2 and Platform Manager 2 only)
Clock/Address/Command	Select this option to fill in the Clock/Address/Command tab to further customize the module.
Data Mask	Select this option to add a data mask port for each DQS group.
Lock/Jitter Sensitivity	Specify the lock/jitter sensitivity for the Master DLL. (MachXO2 and Platform Manager 2 only)
ISI Calibration	When available, choose the ISI calibration type. (LatticeECP3 only)
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 173: DDR_MEM Dialog Box for LatticeECP3, MachXO2, and Platform Manager 2—Clock/Address/Command Tab

Feature	Description
Number of Clocks	Choose the number of clocks.
Number of Clock Enables	Choose the number of bits in the input and output clock enable ports.
Address Width	Choose the width of the address in bits.
Bank Address Width	Choose the width of the bank address in bits.
Number of ODT	Choose the number of bits in the input and output on-die termination (ODT) ports.
Number of Chip Selects	Choose the number of bits in the chip select port.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Use Dedicated DDR Memory Support” on page 1165
- TN1180, *LatticeECP3 High-Speed I/O Interface*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*

Digital CDR

The clock and data recovery logic takes a clock at approximately the same frequency as the incoming data, then produces this same data re-timed with the appropriate clock phase.

Table 174: Digital CDR Dialog Box

Feature	Description
Number of channel groups (3 channels per group)	The 18 channels supported by the Digital CDR are broken up into 6 groups (A, B, C, D, E, F) of 3 channels. By default these three channels share a common output clock (contact the factory for independent operation). This option selects how many 3-channel groups will be used.
Bus width of the channel output	This option selects the size of the data output bus. The output clock will be divided down to match the bus width.
REFCLK Frequency	The frequency of the clock connected to the REFCLK port of the CDR.
CDR Lock Mode	Sync mode is used for systems where there is 0ppm offset between the REFCLK and incoming data rate. Async mode is used for systems where there is an offset between the REFCLK rate and incoming data rate.
REFCLK source routing	This option selects either edge or primary clock routing for the connection to the CDR REFCLK.
GSR Enabled	Enables or disables the GSR from resetting the Digital CDR.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- TN1122, *LattidceSC MACO Core LSCDR1X18 Low-Speed Clock and Data Recovery User's Guide*

Distributed_DPRAM

A dual-port RAM using LUT logic. The module provides a variety of RAM organizations and is implemented as PFU-based, distributed memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

Table 175: Distributed_DPRAM Dialog Box

Feature	Description
Address Depth	The Address Depth entry field is used to specify the address size in terms of the number of words in the ROM. The maximum address depth will vary depending on the device used.
Data Width	The Data Width entry field is used to specify the width of the data bus in bits. The data width will vary depending on the device used.
Enable Output Register	When selected the data outputs of the RAM are registered (separate controls for port A and B). Registering the output ports allows the module to run at a faster clock speed (higher f_{MAX}), but uses more area. Unchecking this box will cause the module to run slower, but will use less area in the device. The Enable Output Register option will add additional ports like RdClockEn, RdClock and Reset for the Output registers.
Memory file	(ECP5, LatticeECP3, LatticeSC/M, LatticeXP2, MachXO2, MachXO3L, and Platform Manager 2 only) Allows you to select a memory initialization (.mem) file for the module. A .mem file is not required. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	(ECP5, LatticeECP3, LatticeSC/M, LatticeXP2, MachXO2, MachXO3L, and Platform Manager 2 only) Allows you to choose Binary, Hex, or Addressed Hex format.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*

- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1306, *CrossLink Memory Usage Guide*

pmi_distributed_dpram

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_distributed_dpram.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_addr_width - 1):0
I	Data	Bus	(pmi_data_width - 1):0
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
I	RdAddress	Bus	(pmi_addr_width - 1):0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_distributed_dpram Definition

```
module pmi_distributed_dpram
  #(parameter pmi_addr_depth = 32,
    parameter pmi_addr_width = 5,
    parameter pmi_data_width = 8,
    parameter pmi_regmode = "reg",
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_distributed_dpram")

    (
    input [(pmi_addr_width-1):0] WrAddress,
    input [(pmi_data_width-1):0] Data,
    input WrClock,
    input WE,
    input WrClockEn,
    input [(pmi_addr_width-1):0] RdAddress,
    input RdClock,
    input RdClockEn,
    input Reset,
```

```
    output [(pmi_data_width-1):0] Q)/* synthesis syn_black_box
*/;

endmodule // pmi_distributed_dpram
```

VHDL pmi_distributed_dpram Definition

```
component pmi_distributed_dpram is
     generic (
       pmi_addr_depth : integer := 32;
       pmi_addr_width : integer := 5;
       pmi_data_width : integer := 8;
       pmi_regmode : string := "reg";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_family : string := "EC";
       module_type : string := "pmi_distributed_dpram"
    );
    port (
     WrAddress : in std_logic_vector((pmi_addr_width-1) downto
0);
     Data : in std_logic_vector((pmi_data_width-1) downto 0);
     WrClock: in std_logic;
     WE: in std_logic;
     WrClockEn: in std_logic;
     RdAddress : in std_logic_vector((pmi_addr_width-1) downto
0);
     RdClock: in std_logic;
     RdClockEn: in std_logic;
     Reset: in std_logic;
     Q : out std_logic_vector((pmi_data_width-1) downto 0)
   );
  end component pmi_distributed_dpram;
```

Parameter Names and Values

The following table contains the pmi_distributed_dpram module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_addr_depth	2 to 8192	32	
pmi_addr_width	1 to 13	5	
pmi_data_width	1 to 256	8	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_init_file	<string>	“none”	The pmi_init_file and pmi_initi_file_format parameters are only supported for LatticeECP3, LatticeSC/M, and LatticeXP2 device families in distributed mode.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_init_file_format	“binary”	“hex”	“binary”	The pmi_init_file and pmi_initi_file_format parameters are only supported for LatticeECP3, LatticeSC/M, and LatticeXP2 device families in distributed mode.
pmi_family	“EC”	“ECP”	“ECP2”	“ECP2M”	“ECP3”	“ECP5U”	“ECP5UM”	“SC”	“SCM”	“XP”	“XP2”	“XO”	“XO2”	“XO3L”	“LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

Distributed_ROM

A read-only memory using LUT logic. The ROM module provides a variety of ROM organizations and is implemented as PFU-based, distributed memory. Ports, properties, and functionality are described in detail in the technical notes in the “See Also” section below.

Table 176: Distributed_ROM Dialog Box

Feature	Description
Address Depth	The Address Depth entry field is used to specify the address size in terms of the number of words in the ROM. The maximum address depth will vary depending on the device used.
Data Width	The Data Width entry field is used to specify the width of the data bus in bits. The maximum data width will vary depending on the device used.
Enable Output Register	When selected, the data outputs of the ROM are registered. Registering the output ports allows the module to run at a faster clock speed (higher f_{MAX}), but uses more area. Unchecking this box will cause the module to run slower, but will use less area in the device. Enable Output Register option will add an additional port like OutClockEn and Reset for the Output registers. The default is checked.
Memory file	Allows you to select a memory initialization (.mem) file for the module. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	Allows you to choose the format of the memory file. Supported formats are Binary, Hex, or Addressed Hex.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.

Table 176: Distributed_ROM Dialog Box (Continued)

Feature	Description
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with FPGA Memories" on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1306, *CrossLink Memory Usage Guide*

pmi_distributed_rom

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_distributed_rom.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	OutClock	Bit	N/A
I	OutClockEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_distributed_rom Definition

```
module pmi_distributed_rom
  #(parameter pmi_addr_depth = 32,
    parameter pmi_addr_width = 5,
    parameter pmi_data_width = 8,
```

```
    parameter pmi_regmode = "reg",
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_distributed_rom")
    (
     input [(pmi_addr_width-1):0] Address,
     input OutClock,
     input OutClockEn,
     input Reset,
     output [(pmi_data_width-1):0] Q)/* synthesis syn_black_box
*/;

endmodule // pmi_distributed_rom
```

VHDL pmi_distributed_rom Definition

```
component pmi_distributed_rom is
     generic (
       pmi_addr_depth : integer := 32;
       pmi_addr_width : integer := 5;
       pmi_data_width : integer := 8;
       pmi_regmode : string := "reg";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_family : string := "EC";
       module_type : string := "pmi_distributed_rom"
    );
    port (
     Address : in std_logic_vector((pmi_addr_width-1) downto
0);
     OutClock: in std_logic;
     OutClockEn: in std_logic;
     Reset: in std_logic;
     Q : out std_logic_vector((pmi_data_width-1) downto 0)
   );
  end component pmi_distributed_rom;
```

Parameter Names and Values

The following table contains the pmi_distributed_rom module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value
pmi_addr_depth	2 to 8192	32
pmi_addr_width	1 to 13	5
pmi_data_width	1 to 128	8
pmi_regmode	“reg” | “noreg”	“reg”
pmi_init_file	<string>	“none”
pmi_init_file_format	“binary” | “hex”	“binary”
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” | “LPTM”	“EC”

See Also

- "Use PMI or IPexpress?" on page 191
- "Using PMI" on page 200

Distributed_SPRAM

A single-port RAM using LUT logic. The RAM module provides a variety of RAM organizations and is implemented as PFU-based, distributed memory. Ports, properties, and functionality are described in detail in the technical notes in the "See Also" section below.

Table 177: Distributed_SPRAM Dialog Box

Feature	Description
Address Depth	The Address Depth entry field is used to specify the address size in terms of the number of words in the ROM. The maximum address depth will vary depending on the device used.
Data Width	The Data Width entry field is used to specify the width of the data bus in bits. The maximum data width will vary depending on the device used.
Enable Output Register	When selected, the data outputs of the RAM are registered. Registering the output ports allows the module to run at a faster clock speed (higher f_{MAX}), but uses more area. Unchecking this box will cause the module to run slower, but will use less area in the device. The Enable Output Register option will add an additional Reset port for the Output registers.
Memory file	(ECP5, LatticeECP3, LatticeSC/M, LatticeXP2, MachXO2, MachXO3L, and Platform Manager 2 only) Allows you to select a memory initialization (.mem) file for the module. A .mem file is not required. See "Creating a Memory Initialization File with Memory Generator" on page 203.
Memory File Format	(ECP5, LatticeECP3, LatticeSC/M, LatticeXP2, MachXO2, MachXO3L, and Platform Manager 2 only) Allows you to choose the format of the memory file. Supported formats are Binary, Hex, or Addressed Hex.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with FPGA Memories" on page 1162

- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1306, *CrossLink Memory Usage Guide*

pmi_distributed_spram

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_distributed_spram.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	Data	Bus	(pmi_data_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	WE	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_distributed_spram Definition

```
module pmi_distributed_spram
  #(parameter pmi_addr_depth = 32,
    parameter pmi_addr_width = 5,
    parameter pmi_data_width = 8,
    parameter pmi_regmode = "reg",
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_distributed_spram")

    (
     input [(pmi_addr_width-1):0] Address,
     input [(pmi_data_width-1):0] Data,
     input Clock,
     input ClockEn,
     input WE,
     input Reset,
```

```
     output [(pmi_data_width-1):0] Q)/* synthesis syn_black_box
*/;

endmodule // pmi_distributed_spram
```

VHDL pmi_distributed_spram Definition

```
component pmi_distributed_spram is
     generic (
       pmi_addr_depth : integer := 32;
       pmi_addr_width : integer := 5;
       pmi_data_width : integer := 8;
       pmi_regmode : string := "reg";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_family : string := "EC";
       module_type : string := "pmi_distributed_spram"
    );
    port (
     Address : in std_logic_vector((pmi_addr_width-1) downto
0);
     Data : in std_logic_vector((pmi_data_width-1) downto 0);
     Clock: in std_logic;
     ClockEn: in std_logic;
     WE: in std_logic;
     Reset: in std_logic;
     Q : out std_logic_vector((pmi_data_width-1) downto 0)
   );
  end component pmi_distributed_spram;
```

Parameter Names and Values

The following table contains the pmi_distributed_spram module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_addr_depth	2 to 8192	32	
pmi_addr_width	1 to 13	5	
pmi_data_width	1 to 128	8	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_init_file	<string>	“none”	The pmi_init_file and pmi_initi_file_format parameters are only supported for LatticeECP3, LatticeSC/M, and LatticeXP2 device families in distributed mode.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_init_file_format	“binary”	“hex”	“binary”	The pmi_init_file and pmi_initi_file_format parameters are only supported for LatticeECP3, LatticeSC/M, and LatticeXP2 device families in distributed mode.
pmi_family	“EC”	“ECP”	“ECP2”	“ECP2M”	“ECP3”	“ECP5U”	“ECP5UM”	“SC”	“SCM”	“XP”	“XP2”	“XO”	“XO2”	”XO3L”	“LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

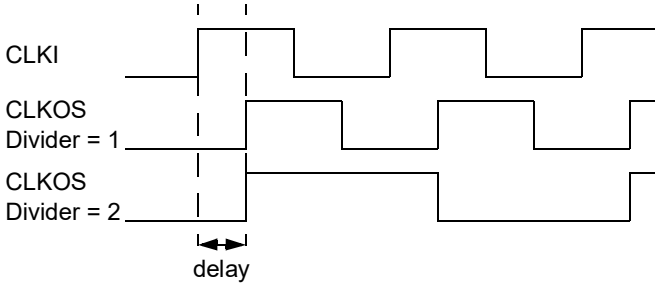
DLL

DLLs (delayed lock loop) are used in a variety of applications such as clock injection removal, clock injection match, and time reference delay, which is used to adjust the DQS strobe delay setting for DDR memory applications.

Table 178: DLL Dialog Box

Feature	Description
Specify the DLL usage mode	By selecting the mode of operation, the generated netlist includes the appropriate library element so that the predefined settings are fixed.
CLKI Frequency	Specify the input frequency of the DLL in MHz. This is passed as an attribute to the netlist and is also used to determine the configuration of the DLL in Clock Injection Delay Removal mode (above).
CLKOS Divider	The frequency of CLKOS is CLKI divided by this value. This also affects the CLKOS phase shift (below) when the DLL usage mode (above) is “Time Reference Delay.”

Table 178: DLL Dialog Box (Continued)

Feature	Description
CLKOS Phase Shift (degree)	Choose the phase shift of the CLKOS output relative to CLKI. Available only when the DLL usage mode (above) is “Time Reference Delay.” **Note:** The amount of the phase shift is based on the period of CLKI, not CLKOS. The delay between CLKI and CLKOS stays the same regardless of changes in the CLKOS Divider, which changes the frequency and period of CLKOS. For example, a Phase Shift of 90 degrees is always 1/4 of the CLKI period regardless of the Divider value and the resulting CLKOS frequency: 
CLKFB	Specify the feedback input. If you choose User Clock, specify the frequency of the feedback clock in terms of CLKI. Not available when the DLL usage mode (above) is “Time Reference Delay.”
Provide SMI ports on DLL module	Select to generate SMI ports. The SMI port can be used to modify the settings of the DLL after configuration of the FPGA. (LatticeSC/M only)
Provide RSTN port	Select to add a reset port. A reset can shut off the output clock and have undesirable effects on the system.
Provide DIFF port	Select to add a DIFF signal that indicates when DCNTL is different from the internal setting and that an update is needed. Only available with LatticeECP3 when the DLL usage mode (above) is “Time Reference Delay.”
Provide GRAYI/O and INCI/O ports	Select to add a gray-coded control bus for other DLLs and an INC port. If the DLL usage mode (above) is “Clock Injection Delay Removal,” these will be input ports. If the mode is “Time Reference Delay,” these will be output ports. (LatticeECP3 only)
Provide DCNTL Ports	Select to add the DCNTL output ports. These ports can be connected to a Delay block (generated as part of a DDR or SDR module, or instantiated by you in the netlist) to match the delays in the DLL. Only available with LatticeECP3 when the DLL usage mode (above) is “Time Reference Delay,” or with LatticeSC/M when the DLL usage mode (above) is “Clock Injection Delay Removal.”
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysCLOCK PLLs and DLLs” on page 1168

- TN1103, *LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide*
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide
- TN1098 - LatticeSC sysCLOCK PLL/DLL User's Guide

DQS

Data strobe used with DDR.

Table 179: DQS Dialog Box

Feature	Description
Number of DQS	Specifies the number of (bidirectional) DQS buffers. Values: 1, 2, 4, 8
Frequency of DQS (MHz)	Specify the frequency of operation of the DLL. The DQS input will be delayed (by T/4) based on the clock frequency. Values: 100, 133, 166, 200
DQS Postamble Solution	Specify if the postamble solution to shut off the DQS clock is to be generated. Values: Yes, No
DDR Tristate Enable for DQS Output	Specify if the tristate enable for the DQS output uses DDR registers. Without this option, the DQS tristate enable is generated without the output DDR registers.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195

Dynamic Bank Controller

Dynamic Bank Controller (INRD and LVDSO) allows you to turn off banks of referenced and differential I/O buffers (used to implement standards such as HSTL, SSTL, and LVDS) These I/Os consume more power than other I/O standards such as LVCMOS. Dynamic Bank Controller (SLEWRATE) allows you to turn off reference current for slewrate slow mode when they are not needed.

RTL or functional simulation is not supported. Only post-place-and-route simulation is supported.

Table 180: Dynamic Bank Controller Dialog Box

Feature	Description
Enable Dynamic InRD Control	Select the referenced and differential input banks to control. ECP5 only supports banks 2, 3, 6, and 7. MachXO2 supports banks 0 to 5 with MachXO2-2000 or larger. MachXO3L supports banks 0 to 5 with MachXO3L-1300 or larger. MachXO3D supports banks 0 to 5. Platform Manager 2 supports banks 0 to 3.
Enable Slew Rate Control (MachXO3D Only)	Select the output banks to control slew rate. MachXO3D supports banks 0 to 5.
Enable Dynamic LVDS Control	Select to control the LVDS output buffers for the bank. MachXO2 only supports bank 0 and only with MachXO2-1200 or larger. MachXO3L only supports bank 0. MachXO3D only supports bank 0. Platform Manager 2 only supports bank 0 and only with LPTM21.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- TN1266, Power Consumption and Management for ECP5 Devices
- TN1198, Power Estimation and Management for MachXO2 Devices

EFB

MachXO2, MachXO3L, MachXO3D, and Platform Manager 2 devices provide embedded hardened functions, such as SPI, I2C, and Timer/Counter. These embedded function blocks (EFB) interface with the user logic through a WISHBONE bus.

The module dialog box has several tabs. Start with the EFB Enables tab to select the function blocks that you want to include in the module. Regardless

of which blocks you select, always set the **Wishbone Clock Frequency** at the bottom of the tab. Then go to the other tabs to set options for the selected blocks.

- EFB Enables, below
- I2C, see Table 182 on page 1450
- SPI, see Table 183 on page 1451
- Timer/Counter, see Table 184 on page 1452
- UFM, see Table 185 on page 1452
- Tamper, see Table 192 on page 1456

Table 181: EFB Dialog Box—EFB Enables Tab

Feature	Description
I2C	The EFB contains three I2C endpoints: Primary Configuration, Primary User, and Secondary User. Select an option to add the ports required to access the specified endpoints. Then set options in the I2C tab (see Table 182 on page 1450).
SPI	Select to add an SPI block to the module. Then set options in the SPI tab (see Table 183 on page 1451).
Timer/Counter	Select to add a timer/counter block to the module. Then select a Timer/Counter Use (see next item) and set options in the Timer/Counter tab (see Table 184 on page 1452).
Timer/Counter Use	Access to the Timer/Counter block through the WISHBONE interface is optional. Using the WISHBONE interface allows you set registers to control the Timer/Counter. To use the WISHBONE interface, select **Dynamic register changes via WISHBONE**. Otherwise, select **Static settings only (no WISHBONE access)**. Then set options in the Timer/Counter tab (see Table 184 on page 1452).
PLL (Dynamic access)	Select to add dynamic access to one or two PLLs. Not available with MachXO2-256 or -640, MachXO3L-640, or Platform Manager 2-20.
Number of PLL's of dynamic access	If you selected PLL (above), select how many PLLs you want to access. The EFB module in MachXO2-4000 and larger and MachXO3L 2100 and larger can access one or two PLLs. Most other devices can access only one PLL.
User Flash Memory	Select to add access to the flash memory through the WISHBONE interface. Then set options in the UFM tab (see Table 185 on page 1452). Not available with MachXO2-256 or MachXO3L.

Table 181: EFB Dialog Box—EFB Enables Tab

Feature	Description
User Flash Memory (MachXO3D Only)	Internal Boot: ▶ Single Boot - Allows you to reserve to boot from Image-0:CFG0 and optionally, include UFM0. ▶ Dual Boot - Allows you to reserve to boot from Image-0:CFG0 and optionally, include UFM0. Also, allows you to reserve to boot from Image-1:CFG1 and optionally, include UFM1. External Boot - Boot from external flash memory. Total available UFM: 33590 pages(537440 bytes) and UFM used in EFB: 6 Pages (96 bytes). ▶ First number is total available UFM. ▶ Second number is the total UFM used as specified in the selected UFM tab(s).
WISHBONE	Select to have a WISHBONE interface without any of the other functions.
WISHBONE Clock Frequency	Specify the frequency of the clock that the module will run on.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 182: EFB Dialog Box—I2C Tab

Feature	Description
General Call Enable	Enables the generic call response in slave mode.
Wakeup Enable	Enables the I2C block to signal the on-chip power manager to wake up the part from standby/sleep mode when the slave address is called. If not selected, the I2C block is powered down during standby/sleep mode.
I2C Bus Performance	Select a frequency for the SCL clock.
Clock Prescale Value	Calculated from the WISHBONE Clock Frequency on the EFB Enables tab and the I2C Bus Performance. This field is read-only.
I2C Addressing	Choose to have a 7- or a 10-bit slave address.
Slave Address	Enter the slave address. The last two bits are fixed. The address of the primary I2C block always ends in 01 (xxxxx01). The address of the secondary I2C block always ends in 10 (xxxxx10).
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.

Table 182: EFB Dialog Box—I2C Tab

Feature	Description
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 183: EFB Dialog Box—SPI Tab

Feature	Description
SPI Mode	For slave mode only, select **Slave**. To dynamically control the mode, select **Slave & Master**.
Master Clock Rate	Enter the desired rate for the clock in MHz. The Actual box shows the actual rate based on the desired value and the WISHBONE clock frequency entered in the EFB Enables tab. Available only for master mode.
LSB First	Select to have the least-significant bit (LSB) first in the data.
Inverted Clock	Select to invert the clock polarity.
Phase Adjust	Select to allow changing the clock/data phase relationship.
Slave Handshake Mode	Select to include the Slave Handshake Mode, which can be useful when the SPI slave is running at high speed. In this mode, the SPI block signals to the external master when data can be sent.
Master Chip Selects	Select the number of chip select outputs desired. Available only for master mode.
Tx Ready	Select to add an interrupt signal that indicates when the SPI transmit data register is empty.
Rx Ready	Select to add an interrupt signal that indicates when the SPI receive data register contains valid data.
Tx Overrun	Select to add an interrupt signal that indicates when the slave SPI select signal is active.
Rx Overrun	Select to add an interrupt signal that indicates when receiving new data before reading the previous data.
Enable Port	Adds an interrupt output signal from the SPI block.
Wakeup Enable	Enables the SPI block to signal the on-chip power manager to wake up the part from low-power modes.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 184: EFB Dialog Box—Timer/Counter Tab

Feature	Description
Timer/Counter Mode	Select the mode: CTCM: Clear timer on compare match WATCHDOG FASTPWM: Fast PWM PFCPWM: Phase and frequency correct PWM
Output Function	Select the type of output.
Clock Edge Selection	Select the clock edge to use.
Use On-Chip Oscillator	Select to use the on-chip oscillator.
Prescale Divider Value	Select a value for the prescale divider.
Enable Interrupt Register	Select conditions that will trigger the interrupt signal. Overflow, Output Compare Match, and Input Capture are only available if “Dynamic register changes via WISHBONE” was selected in the EFB Enables tab. Standalone Overflow is only available if “Static settings only (no WISHBONE access)” was selected in the EFB Enables tab.
Set Top Counter Value	Select to set the maximum counter value.
Timer/Counter Top	Type in a maximum counter value.
Output Compare Value	Enter a value for the output comparison. Not available with the CTCM Timer/Counter mode.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 185: EFB Dialog Box—UFM Tab -- MachXO2, MachXO3L, Platform Manager 2 Only

Feature	Description
Enter Number of Pages(128 bits in one page)	Specify the size of the memory in “pages” with 128 bits/page.
Initialization Data Starts at Page	Read-only field showing the page that the initialization data will start at. This location depends on the number of pages specified above and the size of the flash memory available in the selected device.
User Flash Memory Is Initialized With All 0s	Initializes the memory with zeros instead of a data file (see next item).

Table 185: EFB Dialog Box—UFM Tab -- MachXO2, MachXO3L, Platform Manager 2 Only

Feature	Description
Upload User Flash Memory Initialization Data File	Initializes the memory with a memory initialization (.mem) file that you specify in the text box. Click the browse button next to the box to browse to the file. Below the box, specify whether the file is in hexadecimal or binary format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 186: EFB Dialog Box—UFM_CFG0 Tab -- MachXO3D Only

Feature	Description
Enter Number of Pages(128 bits in one page)	Specify the size of the memory in “pages” with 128 bits/page.
Initialization Data Starts at Page	Read-only field showing the page that the initialization data will start at. This location depends on the number of pages specified above and the size of the flash memory available in the selected device.
User Flash Memory Is Initialized With All 0s	Initializes the memory with zeros instead of a data file (see next item).
Upload User Flash Memory Initialization Data File	Initializes the memory with a memory initialization (.mem) file that you specify in the text box. Click the browse button next to the box to browse to the file. Below the box, specify whether the file is in hexadecimal or binary format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 187: EFB Dialog Box—UFM0 Tab -- MachXO3D Only

Feature	Description
Enter Number of Pages(128 bits in one page)	Specify the size of the memory in “pages” with 128 bits/page.
Initialization Data Starts at Page	Read-only field showing the page that the initialization data will start at. This location depends on the number of pages specified above and the size of the flash memory available in the selected device.
User Flash Memory Is Initialized With All 0s	Initializes the memory with zeros instead of a data file (see next item).

Table 187: EFB Dialog Box—UFM0 Tab -- MachXO3D Only

Feature	Description
Upload User Flash Memory Initialization Data File	Initializes the memory with a memory initialization (.mem) file that you specify in the text box. Click the browse button next to the box to browse to the file. Below the box, specify whether the file is in hexadecimal or binary format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 188: EFB Dialog Box—UFM_CFG1 Tab -- MachXO3D Only

Feature	Description
Enter Number of Pages(128 bits in one page)	Specify the size of the memory in “pages” with 128 bits/page.
Initialization Data Starts at Page	Read-only field showing the page that the initialization data will start at. This location depends on the number of pages specified above and the size of the flash memory available in the selected device.
User Flash Memory Is Initialized With All 0s	Initializes the memory with zeros instead of a data file (see next item).
Upload User Flash Memory Initialization Data File	Initializes the memory with a memory initialization (.mem) file that you specify in the text box. Click the browse button next to the box to browse to the file. Below the box, specify whether the file is in hexadecimal or binary format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 189: EFB Dialog Box—UFM1 Tab -- MachXO3D Only

Feature	Description
Enter Number of Pages(128 bits in one page)	Specify the size of the memory in “pages” with 128 bits/page.
Initialization Data Starts at Page	Read-only field showing the page that the initialization data will start at. This location depends on the number of pages specified above and the size of the flash memory available in the selected device.
User Flash Memory Is Initialized With All 0s	Initializes the memory with zeros instead of a data file (see next item).

Table 189: EFB Dialog Box—UFM1 Tab -- MachXO3D Only

Feature	Description
Upload User Flash Memory Initialization Data File	Initializes the memory with a memory initialization (.mem) file that you specify in the text box. Click the browse button next to the box to browse to the file. Below the box, specify whether the file is in hexadecimal or binary format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 190: EFB Dialog Box—UFM2 Tab -- MachXO3D Only

Feature	Description
Enter Number of Pages(128 bits in one page)	Specify the size of the memory in “pages” with 128 bits/page.
Initialization Data Starts at Page	Read-only field showing the page that the initialization data will start at. This location depends on the number of pages specified above and the size of the flash memory available in the selected device.
User Flash Memory Is Initialized With All 0s	Initializes the memory with zeros instead of a data file (see next item).
Upload User Flash Memory Initialization Data File	Initializes the memory with a memory initialization (.mem) file that you specify in the text box. Click the browse button next to the box to browse to the file. Below the box, specify whether the file is in hexadecimal or binary format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 191: EFB Dialog Box—UFM3 Tab -- MachXO3D Only

Feature	Description
Enter Number of Pages(128 bits in one page)	Specify the size of the memory in “pages” with 128 bits/page.
Initialization Data Starts at Page	Read-only field showing the page that the initialization data will start at. This location depends on the number of pages specified above and the size of the flash memory available in the selected device.
User Flash Memory Is Initialized With All 0s	Initializes the memory with zeros instead of a data file (see next item).

Table 191: EFB Dialog Box—UFM3 Tab -- MachXO3D Only

Feature	Description
Upload User Flash Memory Initialization Data File	Initializes the memory with a memory initialization (.mem) file that you specify in the text box. Click the browse button next to the box to browse to the file. Below the box, specify whether the file is in hexadecimal or binary format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 192: EFB Dialog Box - Tamper Tab -- MachXO3D Only

Feature	Description
Password Protection	Enables the detection of the unexpected access to overwrite password.
Locked Flash or SRAM	Enables the detection of actions that try to access a locked flash sector or SRAM.
Manufacture Mode	Enables the detection of entering manufacturing mode.
JTAG Port	Enables the threat detection for JTAG port.
SSPI Port	Enables the threat detection for SSPI port.
SI2C Port	Enables the threat detection for SI2C port.
WB Port	Enables the threat detection for WB port.
Port Lock	Enables the port lock function.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- TN1205. Using User Flash Memory and Hardened Control Functions in MachXO2 Devices
- TN1246 - Using User Flash Memory and Hardened Control Functions in MachXO2 Devices Reference Guide
- TN1293 - Using Hardened Control Functions in MachXO3L Devices
- TN1294 - Using Hardened Control Functions in MachXO3L Devices Reference Guide

EXTREF

External reference clock module.

Table 193: EXTREF Dialog Box

Feature	Description
Termination Resistance	Choose a value for the termination resistance.
DC Bias Enabled	Select to include a DC bias.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- TN1261, ECP5 SERDES/PCS Usage Guide

FFT_Butterfly

Butterfly module for the popular Cooley-Tukey Fast Fourier Transform (FFT), which recursively breaks down the discrete Fourier transform (DTF) into smaller DTFs. The ultimate computation unit is the FFT_Butterfly module. This module offers radix-2 or radix-4 and decimation in time or decimation in frequency algorithms.

Table 194: FFT_Butterfly Dialog Box

Feature	Description
FFT Butterfly Mode	Specify the algorithm. Radix-2 and radix-4 are supported. For each, the implementation can be decimation in time (DIT) or decimation in frequency (DIF).
Use DSP Block	Select Use DSP block to implement the complex multiplier. This greatly increases performance and reduces the number of LUTs used. When this option is not selected, the multiplier is implemented in SLICE logic. This option is only available for architectures that have DSP blocks.
Treat Inputs as Complex Numbers	Specifies that the inputs are complex numbers and require the full complex multiplier to compute the results. If selected, both real and imaginary ports are used and it requires three multipliers and some adders to implement this function. If this option is not selected, both real and imaginary ports will still be generated; however, the imaginary ports will not be used. Two multipliers will be sufficient to implement this function.
Input Data Bit Width	Specify the width of the input data.
Twiddle Factor Bit Width	Specify the width of the sin-cos twiddle factors.

Table 194: FFT_Butterfly Dialog Box (Continued)

Feature	Description
Output Bit Width	Shows the bit width of each of the result buses. The output width is the sum of the input data and twiddle factor widths.
Specify the Number of Pipeline Stages	Specify the number of register stages inserted. More stages improves achievable frequency of operation, but the latency for the output increases.
Enable Input Registers	Specify if input registers are required.
Enable Output Registers	Specify if output registers are required.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195

FIFO

A FIFO using EBR (Embedded Block RAM). The FIFO module provides a variety of FIFO memory organizations. This FIFO has only one clock. Binary counters are used in the generated netlist. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

Table 195: FIFO Dialog Box

Feature	Description
FIFO Implementation	**EBR Based** (or **Only**) - Contains Embedded Block RAM implementation. **LUT Based** (or **Only**) – Contains LUTs and logic implementation. The LUT implementation has some limitations: ▶ The maximum address depth is 8192. ▶ The Enable ECC option (below) is not available.
Address_depth	Specifies the depth of the FIFO.
Data_width	Specifies the data width of the FIFO.
Enable Output Register	Specify that registers are used for the output data. There is no change to the flag logic.

Table 195: FIFO Dialog Box (Continued)

Feature	Description
Controlled by RdEn	Specify that the registers for the output data are controlled by the read enable. If the read enable is directly controlled by the Empty output, this will not release the final data. When this option is not selected, the output registers are always enabled. In this case, the final data will come out even when read enable is low. To enable this option, select **Enable Output Register**.
Flag Control	**Almost Empty Flag -** Specifies the threshold value for the partial empty flag. **Almost Full Flag -** Specifies the threshold value for the partial full flag. In the drop-down menus, choose the threshold options for each flag. Static thresholds are set at the time of module generation. Dynamic thresholds have additional ports to allow changing the thresholds during operation. A single threshold allows specifying one value to be used for both assert and deassert. This uses fewer resources, but has a latency for deassert. Dual thresholds allow different values to assert and deassert the flag. If you chose a static threshold, specify the Assert and, if needed, Deassert values.
Reset Mode	(ECP5 only) Asynchronous or synchronous reset supported. Reset release requires that reset assertion be asynchronous.
Data Count Options	Enables the data count output. This output is synchronized with Clock and shows the number of words in the FIFO. Since it is synchronized with the write clock, the changes due to reads will show a few cycles later.
Enable ECC (not supported for Data Width > 64)	Allows error correction of single errors and error detection of double errors. This is only available with the EBR implementation and is not supported for data widths greater than 64 bits.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*

- TN1137, *LatticeXP2 Memory Usage Guide*

pmi_fifo

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_fifo.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width - 1):0
I	Clock	Bit	N/A
I	WrEn	Bit	N/A
I	RdEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0
O	Empty	Bit	N/A
O	Full	Bit	N/A
O	AlmostEmpty	Bit	N/A
O	AlmostFull	Bit	N/A

Verilog pmi_fifo Definition

```
module pmi_fifo #(
         parameter pmi_data_width = 8,
         parameter pmi_data_depth = 256,
         parameter pmi_full_flag = 256,
         parameter pmi_empty_flag = 0,
         parameter pmi_almost_full_flag = 252,
         parameter pmi_almost_empty_flag = 4,
         parameter pmi_regmode = "reg",
         parameter pmi_family = "EC" ,
         parameter module_type = "pmi_fifo",
        parameter pmi_implementation = "LUT")

  (input  [pmi_data_width-1:0] Data,
   input Clock,
   input WrEn,
   input RdEn,
   input Reset,
   output [pmi_data_width-1:0] Q,
   output Empty,
   output Full,
   output AlmostEmpty,
   output AlmostFull)/*synthesis syn_black_box */;

endmodule // pmi_fifo
```

VHDL pmi_fifo Definition

```
component pmi_fifo is
     generic (
       pmi_data_width : integer := 8;
       pmi_data_depth : integer := 256;
       pmi_full_flag : integer := 256;
       pmi_empty_flag : integer := 0;
       pmi_almost_full_flag : integer := 252;
       pmi_almost_empty_flag : integer := 4;
       pmi_regmode : string := "reg";
       pmi_family : string := "EC" ;
       module_type : string := "pmi_fifo";
       pmi_implementation : string := "LUT"
    );
    port (
     Data : in std_logic_vector(pmi_data_width-1 downto 0);
     Clock: in std_logic;
     WrEn: in std_logic;
     RdEn: in std_logic;
     Reset: in std_logic;
     Q : out std_logic_vector(pmi_data_width-1 downto 0);
     Empty: out std_logic;
     Full: out std_logic;
     AlmostEmpty: out std_logic;
     AlmostFull: out std_logic
   );
  end component pmi_fifo;
```

Parameter Names and Values

The following table contains the pmi_fifo module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value
pmi_addr_depth	2 to 8192	256
pmi_data_width	1 to 256	8
pmi_almost_empty	1 to 512	4
pmi_almost_full	1 to 512	252
pmi_full_flag	1 to pmi_data_depth	256
pmi_empty_flag	0	0
pmi_regmode	“reg” | “noreg”	“reg”
pmi_implementation	“EBR” | “LUT”	“EBR”
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2”	“EC”

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

FIFO_DC

A dual-clock FIFO using EBR (Embedded Block RAM). The FIFO_DC module provides a variety of FIFO memory organizations. This FIFO has two clocks. The generated netlist has gray counters to handle clock domain transfer. Ports, properties, and functionality are described in detail in the “See Also” section below.

The options vary with the device family. See:

- “FIFO_DC Options for ECP5” on page 1463
- “FIFO_DC Options for LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeXP2)” on page 1465
- “FIFO_DC Options for LatticeSC/M, MachXO” on page 1467
- “FIFO_DC Options for MachXO2, MachXO3L, Platform Manager 2” on page 1469
- “FIFO_DC Options for Platform Manager” on page 1471

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1306, *CrossLink Memory Usage Guide*

FIFO_DC Options for ECP5

A dual-clock FIFO using EBR (Embedded Block RAM). The FIFO_DC module provides a variety of FIFO memory organizations. This FIFO has two clocks. The generated netlist has gray counters to handle clock domain transfer. Ports, properties, and functionality are described in detail in the “See Also” section below.

Table 196: FIFO_DC Dialog Box for ECP5

Feature	Description
FIFO Implementation	**EBR Only** – Contains only Embedded Block RAM implementation. **LUT Only** – Contains only LUT implementation. The LUT Only implementation has some limitations: ▶ The maximum address depth is 8192. ▶ The Read Address Depth is the same as for Write. ▶ ECC is not available.
Write	Specify the number of address locations and the data width for the input, Data. The Write Data Width is also the Read Data Width. With LUT Only implementations, the Write Address Depth is also the Read Address Depth.
Read	Specify the number of address locations and the data width for the output, Q. The Read Data Width is the same as the Write Data Width. With LUT Only implementations, the Read Address Depth is the same as the Write Address Depth.
Enable Output Register	Specify that registers are used for the output data. There is no change to the flag logic.
Controlled by RdEn	Specify that the registers for the output data are controlled by the read enable. Available only if Enable Output Register is selected. If the read enable is directly controlled by the Empty output, this will not release the final data. When this option is not selected, the output registers are always enabled. In this case, the final data will come out even when read enable is low.
Flag Control	**Almost Empty Flag -** Select to include an almost empty flag. **Almost Full Flag -** Select to include an almost full flag. In the drop-down menus, choose the threshold options for each flag. Static thresholds are set at the time of module generation. Dynamic thresholds have additional ports to allow changing the thresholds during operation. A single threshold allows specifying one value to be used for both assert and deassert. This uses fewer resources, but has a latency for deassert. Dual thresholds allow different values to assert and deassert the flag. If you chose a static threshold, specify the Assert and, if needed, Deassert values.
Reset Mode	Select whether the reset is asynchronous or synchronous. Reset release requires that reset assertion be asynchronous.

Table 196: FIFO_DC Dialog Box for ECP5 (Continued)

Feature	Description
Data Count Options	**Data Count (Synchronized with Write Clock) -** Enables the data count output. This output is synchronized with WrClock and shows the number of words in the FIFO. Since it is synchronized with the write clock, the changes due to reads will show a few cycles later. **Data Count (Synchronized with Read Clock) -** Enables the data count output. This output is synchronized with RdClock and shows the number of words in the FIFO. Since it is synchronized with the read clock, the changes due to writes will show a few cycles later.
Enable ECC	Allows error correction of single errors and error detection of double errors. This is not supported for data widths greater than 64 bits. (Not available with LUT Only implementation.)
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*

FIFO_DC Options for LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeXP2)

A dual-clock FIFO using EBR (Embedded Block RAM). The FIFO_DC module provides a variety of FIFO memory organizations. This FIFO has two clocks. The generated netlist has gray counters to handle clock domain transfer. Ports, properties, and functionality are described in detail in the “See Also” section below.

Table 197: FIFO_DC Dialog Box

Feature	Description
FIFO Implementation	**EBR Based** – Contains Embedded Block RAM implementation. **LUT Based** – Contains LUTs and logic implementation. The LUT Based implementation has some limitations: ▶ The maximum address depth is 8192. ▶ The Read Address Depth is the same as for Write. ▶ ECC is not available.
Write	Specify the number of address locations and the data width for the input, Data. The Write Data Width is also the Read Data Width. With LUT Based implementations, the Write Address Depth is also the Read Address Depth.
Read	Specify the number of address locations and the data width for the output, Q. The Read Data Width is the same as the Write Data Width. With LUT Based implementations, the Read Address Depth is the same as the Write Address Depth.
Enable Output Register	Specify that registers are used for the output data. There is no change to the flag logic.
Controlled by RdEn	Specify that the registers for the output data are controlled by the read enable. Available only if Enable Output Register is selected. If the read enable is directly controlled by the Empty output, this will not release the final data. When this option is not selected, the output registers are always enabled. In this case, the final data will come out even when read enable is low.
Flag Control	**Almost Empty Flag -** Select to include an almost empty flag. **Almost Full Flag -** Select to include an almost full flag. In the drop-down menus, choose the threshold options for each flag. Static thresholds are set at the time of module generation. Dynamic thresholds have additional ports to allow changing the thresholds during operation. A single threshold allows specifying one value to be used for both assert and deassert. This uses fewer resources, but has a latency for deassert. Dual thresholds allow different values to assert and deassert the flag. If you chose a static threshold, specify the Assert and, if needed, Deassert values.

Table 197: FIFO_DC Dialog Box (Continued)

Feature	Description
Data Count Options	**Data Count (Synchronized with Write Clock) -** Enables the data count output. This output is synchronized with WrClock and shows the number of words in the FIFO. Since it is synchronized with the write clock, the changes due to reads will show a few cycles later. **Data Count (Synchronized with Read Clock) -** Enables the data count output. This output is synchronized with RdClock and shows the number of words in the FIFO. Since it is synchronized with the read clock, the changes due to writes will show a few cycles later.
Enable ECC	Allows error correction of single errors and error detection of double errors. This is not supported for data widths greater than 64 bits. (Not available with LUT Based implementation.)
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1137, *LatticeXP2 Memory Usage Guide*

FIFO_DC Options for LatticeSC/M, MachXO

A dual-clock FIFO using EBR (Embedded Block RAM). The FIFO_DC module provides a variety of FIFO memory organizations. This FIFO has two clocks. The generated netlist has gray counters to handle clock domain transfer. Ports, properties, and functionality are described in detail in the “See Also” section below.

Table 198: FIFO_DC Dialog Box for LatticeSC/M and MachXO

Feature	Description
FIFO Implementation	**EBR Only** – Contains only Embedded Block RAM implementation. The EBR Only implementation has some limitations: ▶ Almost Empty and Almost Full flags are always included. ▶ Data Count Options are not available. **LUT Based** – Contains LUTs and logic implementation. The LUT Based implementation has some limitations: ▶ The maximum address depth is 8192. ▶ The Read Address Depth is the same as for Write. ▶ ECC is not available.
Write	Specify the number of address locations and the data width for the input, Data. The Write Data Width is also the Read Data Width. With LUT Based implementations, the Write Address Depth is also the Read Address Depth.
Read	Specify the number of address locations and the data width for the output, Q. The Read Data Width is the same as the Write Data Width. With LUT Based implementations, the Read Address Depth is the same as the Write Address Depth.
Enable Output Register	Specify that registers are used for the output data. There is no change to the flag logic.
Controlled by RdEn	Specify that the registers for the output data are controlled by the read enable. Available only if Enable Output Register is selected. If the read enable is directly controlled by the Empty output, this will not release the final data. When this option is not selected, the output registers are always enabled. In this case, the final data will come out even when read enable is low.
Flag Control	**Almost Empty Flag -** Select to include an almost empty flag. With EBR Only implementation, the flag is automatically included. **Almost Full Flag -** Select to include an almost full flag. With EBR Only implementation, the flag is automatically included. In the drop-down menus, choose the threshold options for each flag. Static thresholds are set at the time of module generation. Dynamic thresholds have additional ports to allow changing the thresholds during operation. A single threshold allows specifying one value to be used for both assert and deassert. This uses fewer resources, but has a latency for deassert. Dual thresholds allow different values to assert and deassert the flag. If you chose a static threshold, specify the Assert and, if needed, Deassert values.

Table 198: FIFO_DC Dialog Box for LatticeSC/M and MachXO (Continued)

Feature	Description
Reset Mode	Select whether the reset is asynchronous or synchronous.
Data Count Options	**Write Data Count (synchronized with Clock) -** Enables the data count output. This output is synchronized with WrClock and shows the number of words in the FIFO. Since it is synchronized with the write clock, the changes due to reads will show a few cycles later. (Not available with EBR Only implementation.) **Read Data Count (synchronized with Clock) -** Enables the data count output. This output is synchronized with RdClock and shows the number of words in the FIFO. Since it is synchronized with the read clock, the changes due to writes will show a few cycles later. (Not available with EBR Only implementation.)
Enable ECC	Allows error correction of single errors and error detection of double errors. This is not supported for data widths greater than 64 bits. (Not available with LUT Based implementation.)
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1092, *Memory Usage Guide for MachXO2 Devices*

FIFO_DC Options for MachXO2, MachXO3L, Platform Manager 2

A dual-clock FIFO using EBR (Embedded Block RAM). The FIFO_DC module provides a variety of FIFO memory organizations. This FIFO has two clocks. The generated netlist has gray counters to handle clock domain transfer. Ports, properties, and functionality are described in detail in the “See Also” section below.

Table 199: FIFO_DC Dialog Box for MachXO2, MachXO3L, and Platform Manager 2

Feature	Description
FIFO Implementation	**EBR Only** – (With MachXO2, requires 640 LUTs or larger.) Contains only Embedded Block RAM implementation. The EBR Only implementation has some limitations: ▶ Almost Empty and Almost Full flags are always included. ▶ Data Count Options are not available. **LUT Based** – Contains LUTs and logic implementation. The LUT Based implementation has some limitations: ▶ The maximum address depth is 8192. ▶ The Read Address Depth is the same as for Write. ▶ ECC is not available.
Write	Specify the number of address locations and the data width for the input, Data. The Write Data Width is also the Read Data Width. With LUT Based implementations, the Write Address Depth is also the Read Address Depth.
Read	Specify the number of address locations and the data width for the output, Q. The Read Data Width is the same as the Write Data Width. With LUT Based implementations, the Read Address Depth is the same as the Write Address Depth.
Enable Output Register	Specify that registers are used for the output data. There is no change to the flag logic.
Controlled by RdEn	Specify that the registers for the output data are controlled by the read enable. Available only if Enable Output Register is selected. If the read enable is directly controlled by the Empty output, this will not release the final data. When this option is not selected, the output registers are always enabled. In this case, the final data will come out even when read enable is low.
Enable Output ClockEn	Adds an output clock enable port. This is an input port. Requires that the Output Register (above) be enabled first.

Table 199: FIFO_DC Dialog Box for MachXO2, MachXO3L, and Platform Manager 2 (Continued)

Feature	Description
Flag Control	**Almost Empty Flag -** Select to include an almost empty flag. With EBR Only implementation, the flag is automatically included. **Almost Full Flag -** Select to include an almost full flag. With EBR Only implementation, the flag is automatically included. In the drop-down menus, choose the threshold options for each flag. Static thresholds are set at the time of module generation. Dynamic thresholds have additional ports to allow changing the thresholds during operation. A single threshold allows specifying one value to be used for both assert and deassert. This uses fewer resources, but has a latency for deassert. Dual thresholds allow different values to assert and deassert the flag. If you chose a static threshold, specify the Assert and, if needed, Deassert values.
Reset Mode	Select whether the reset is asynchronous or synchronous. The Output Register (above) must be enabled first. Reset release requires that reset assertion be asynchronous.
Data Count Options	**Data Count (Synchronized with Write Clock) -** Enables the data count output. This output is synchronized with WrClock and shows the number of words in the FIFO. Since it is synchronized with the write clock, the changes due to reads will show a few cycles later. (Not available with EBR Only implementation.) **Data Count (Synchronized with Read Clock) -** Enables the data count output. This output is synchronized with RdClock and shows the number of words in the FIFO. Since it is synchronized with the read clock, the changes due to writes will show a few cycles later. (Not available with EBR Only implementation.)
Enable ECC	Allows error correction of single errors and error detection of double errors. This is not supported for data widths greater than 64 bits.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1290, *Memory Usage Guide for MachXO3L Devices*

FIFO_DC Options for Platform Manager

A dual-clock FIFO using EBR (Embedded Block RAM). The FIFO_DC module provides a variety of FIFO memory organizations. This FIFO has two clocks. The generated netlist has gray counters to handle clock domain transfer. Ports, properties, and functionality are described in detail in the “See Also” section below.

Table 200: FIFO_DC Dialog Box for Platform Manager

Feature	Description
FIFO Implementation	**EBR Only** – Not available with Platform Manager. **LUT Based** – Contains LUTs and logic implementation.
Write	Specify the number of address locations and the data width for the input, Data. The Write and Read Data Widths and Address Depths are the same.
Read	Specify the number of address locations and the data width for the output, Q. The Read Data Width is the same as the Write Data Width. The Read Address Depth is the same as the Write Address Depth.
Enable Output Register	Specify that registers are used for the output data. There is no change to the flag logic.
Controlled by RdEn	Specify that the registers for the output data are controlled by the read enable. Available only if Enable Output Register is selected. If the read enable is directly controlled by the Empty output, this will not release the final data. When this option is not selected, the output registers are always enabled. In this case, the final data will come out even when read enable is low.
Flag Control	**Almost Empty Flag -** Select to include an almost empty flag. **Almost Full Flag -** Select to include an almost full flag. In the drop-down menus, choose the threshold options for each flag. Static thresholds are set at the time of module generation. Dynamic thresholds have additional ports to allow changing the thresholds during operation. A single threshold allows specifying one value to be used for both assert and deassert. This uses fewer resources, but has a latency for deassert. Dual thresholds allow different values to assert and deassert the flag. If you chose a static threshold, specify the Assert and, if needed, Deassert values.
Reset Mode	Select whether the reset is asynchronous or synchronous.
Data Count Options	**Write Data Count (synchronized with Clock) -** Enables the data count output. This output is synchronized with WrClock and shows the number of words in the FIFO. Since it is synchronized with the write clock, the changes due to reads will show a few cycles later. **Read Data Count (synchronized with Clock) -** Enables the data count output. This output is synchronized with RdClock and shows the number of words in the FIFO. Since it is synchronized with the read clock, the changes due to writes will show a few cycles later.
Enable ECC	Not available with Platform Manager.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.

Table 200: FIFO_DC Dialog Box for Platform Manager (Continued)

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with FPGA Memories" on page 1162

pmi_fifo_dc

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_fifo_dc.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width_w - 1):0
I	WrClock	Bit	N/A
I	RdClock	Bit	N/A
I	WrEn	Bit	N/A
I	RdEn	Bit	N/A
I	Reset	Bit	N/A
I	RPReset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0
O	Empty	Bit	N/A
O	Full	Bit	N/A
O	AlmostEmpty	Bit	N/A
O	AlmostFull	Bit	N/A

Verilog pmi_fifo_dc Definition

```
module pmi_fifo_dc #(
        parameter pmi_data_width_w = 18,
        parameter pmi_data_width_r = 18,
        parameter pmi_data_depth_w = 256,
```

```
            parameter pmi_data_depth_r = 256,
            parameter pmi_full_flag = 256,
            parameter pmi_empty_flag = 0,
            parameter pmi_almost_full_flag = 252,
            parameter pmi_almost_empty_flag = 4,
            parameter pmi_regmode = "reg",
            parameter pmi_resetmode = "async",
            parameter pmi_family = "EC" ,
            parameter module_type = "pmi_fifo_dc",
           parameter pmi_implementation = "EBR"
            )

  (input  [pmi_data_width_w-1:0] Data,
   input WrClock,
   input RdClock,
   input WrEn,
   input RdEn,
   input Reset,
   input RPReset,
   output [pmi_data_width_r-1:0] Q,
   output Empty,
   output Full,
   output AlmostEmpty,
   output AlmostFull)/*synthesis syn_black_box */;

endmodule // pmi_fifo_dc
```

VHDL pmi_fifo_dc Definition

```
component pmi_fifo_dc is
     generic (
       pmi_data_width_w : integer := 18;
       pmi_data_width_r : integer := 18;
       pmi_data_depth_w : integer := 256;
       pmi_data_depth_r : integer := 256;
       pmi_full_flag : integer := 256;
       pmi_empty_flag : integer := 0;
       pmi_almost_full_flag : integer := 252;
       pmi_almost_empty_flag : integer := 4;
       pmi_regmode : string := "reg";
       pmi_resetmode : string := "async";
       pmi_family : string := "EC" ;
       module_type : string := "pmi_fifo_dc";
       pmi_implementation : string := "EBR"

    );
    port (
     Data : in std_logic_vector(pmi_data_width_w-1 downto 0);
     WrClock: in std_logic;
     RdClock: in std_logic;
     WrEn: in std_logic;
     RdEn: in std_logic;
     Reset: in std_logic;
     RPReset: in std_logic;
     Q : out std_logic_vector(pmi_data_width_r-1 downto 0);
     Empty: out std_logic;
     Full: out std_logic;
     AlmostEmpty: out std_logic;
     AlmostFull: out std_logic
```

```
    );
end component pmi_fifo_dc;
```

Parameter Names and Values

The following table contains the pmi_fifo_dc module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_data_depth_w	2 to 131072	256	The device depth for the LUT based FIFO_DC range from 2 to 8192. The LatticeECP2/M, LatticeECP3, LatticeSC/M, and LatticeXP2 device family data depths range from 2 to 131072. The LatticeECP/EC and LatticeXP data depths range from 2 to 65536. The MachXO and Platform Manager data depths range from 2 to 16384. The LatticeSC/M, Platform Manager, and MachXO device families support different read/write depths and different read/write data widths.
pmi_data_depth_r	2 to 131072	256	The device depth for the LUT based FIFO_DC range from 2 to 8192. The LatticeECP2/M, LatticeECP3, LatticeSC/M, and LatticeXP2 device family data depths range from 2 to 131072. The LatticeECP/EC and LatticeXP data depths range from 2 to 65536. The MachXO and Platform Manager data depths range from 2 to 16384. The LatticeSC/M, Platform Manager, and MachXO device families support different read/write depths and different read/write data widths.
pmi_data_width_w	1 to 256	18	
pmi_data_width_r	1 to 256	18	
pmi_full_flag	1 to pmi_data_depth_r	256	
pmi_empty_flag	0	0	
pmi_almost_full_flag	1 to pmi_data_depth_w	252	
pmi_almost_empty_flag	1 to pmi_data_depth_r	4	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_resetmode	“async” | “sync”	“async”	The LatticeSC/M and MachXO device families support the pmi_resetmode parameter. The pmi_resetmode parameter applies to the EBR memory implementation.
pmi_implementation	“EBR” | “LUT”	“EBR”	
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” |“LPTM”	“EC”	

See Also

- "Use PMI or IPexpress?" on page 191
- "Using PMI" on page 200

GDDR_7:1

ECP5, MachXO2, MachXO3L, and Platform Manager 2 devices have built-in 7:1 LVDS video interface support. The 7:1 LVDS is a source-synchronous interface for video applications that serialize the 7 bits of parallel data for each cycle of the low-speed clock.

Details of the 7:1 LVDS applications can be found in the MachXO2 Display Interface reference design (RD1093). For more information and to get the source code, go to the Lattice Web site at:

Products > Design Software & Intellectual Property > Intellectual Property > 7:1 LVDS Video Interface

A source code license agreement is required before accessing the source code.

Table 201: GDDR_7:1 Dialog Box

Feature	Description
Interface Type	Specify whether the DDR is for transmit or receive.
Data (or Bus) Width	Specify the width of the data bus in bits.
High-Speed Clock Frequency (pixel clock)	Specify the desired clock frequency in megahertz.
Enable Bit Alignment & Word Alignment Soft IP	Select to add bit and word alignment. Available only with ECP5 and the Receive interface type.
Reference Clock From I/O Pin	Select if the reference clock will come from an I/O pin. Then specify the Reference Clock Input Buffer Type (below). Available only with ECP5 and the Transmit interface type.
Reference Clock Input Buffer Type	Choose the buffer for the input reference clock when the clock is coming from an I/O pin. Available only with ECP5 and the Transmit interface type.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Use Dedicated DDR Memory Support" on page 1165

- TN1265, *ECP5 High-Speed I/O Interface*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1261, ECP5 SERDES/PCS Usage Guide
- TN1293 - Using Hardened Control Functions in MachXO3L Devices

I2C

A System Bus Interface (SBI) or FIFO Interface connects the Hard IP to the fabric. The system bus is connected to two I^2C Hard IPs. Select an option in the first tab to enable the other two tabs.

Table 202: I2C Dialog Box—Hard IP Enables Tab

Feature	Description
Enable Hard Users I2C0	Enables the I2C0 Settings and Interrupts.
System Bus Clock Frequency	Configures the divider settings of the I2C0.
Enable Hard Users I2C1	Enables the I2C1 Settings and Interrupts.
System Bus Clock Frequency	Configures the divider settings of the I2C1.
Configure	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 203: I2C Dialog Box—I2C Settings Tab

Feature	Description
General	**General Call Enable** – Enables I2C General call support. Sets a register based on the active mode (FIFO or System bus). **Include IO Buffers** – Include buffers for the I2C pins. The pins can be placed in any IO location, and will default to the IO LVCMOS standard. Includes the I2C pins I2Cx_SCL and I2Cx_SDA. **SDA Input Delay (ns)** – Add a 50 ns input delay. **SDA Output Delay (ns)** – Add a 50, 75, 125, 150, 200, 300, or 350 ns output delay.
Master Clock Rate	**Desired(KHz)** – Enter a desired clock rate in KHz to automatically determine a divider value and generate a clock close to this value based on the input clock. The frequency of this input System Bus clock is specified in the Hard IP Enables tab. The divider is rounded to the nearest integer after dividing the input System Bus clock by the value entered in this field. **Actual(KHz)** – Since the input System Bus clock cannot always be divided into the user's exact desired master clock, this field displays the actual value. This read-only value is calculated as follows: `Actual clock = rounded(System Bus clock / rounded(System Bus clock / Master clock) )` Divider values exactly halfway between two integers are rounded up (for example, a divider value of 108.5 is rounded to 109).
FIFO Mode	**Enable FIFO Mode** – Allows you change all bits of the I2C address. **CLK Stretch Enable** – Enables data setup time. **RxFIFO Almost Full** – Shows the actual almost full value (not the offset), determined by I2CFIFOTHRESHOLD[9:5] = 32 - I2C_FIFO_RxAlmostF. **TxFIFO Almost Empty** – Shows the actual almost empty value.
I2C Addressing	**Addressing mode** – Select 7-bit or 10-bit addressing. **Address** – Define the Hard I2C address. FIFO mode allows all bits to be programmed, but register mode fixes the lower two bits.
PMU Wake Options	**Address Match** – Wake up the PMU on Address Match. **RxFIFO Almost Full** – Only available for I2C0, as that is the only I2C block connected to the PMU. Enabling this option also enables the Glitchless Clock Switch.
Configure	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 204: I2C Dialog Box—I2C Interrupts Tab

Feature	Description
System Bus Interrupts	**Arbitration Lost** – This indicates when the master has lost its arbitration in master mode. **Tx/Rx Ready** – Enables Tx/Rx Ready status. **Overrun or NACK** – Enables Overrun or NACK status. **General Call IRQ Enable** – Enables the generic call response.
FIFO Interrupts	**General Call** – Enables I2C General call support. Sets a register based on the active mode (FIFO or System bus). **Receive NACK** – Allows the PMU to receive a NACK signal. **Master Read Complete** – Enables Master Read Complete. **Arbitration Lost** – This indicates when the master has lost its arbitration in master mode. **Tx FIFO Sync** – Enables Tx FIFO Sync. **Tx FIFO Underflow** – Enables Tx FIFO Underflow. **Rx FIFO Overflow** – Enables Tx FIFO Overflow.
Configure	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195

LFSR

A linear feedback shift register (LFSR).

Table 205: LFSR Dialog Box

Feature	Description
LFSR Type	Specifies whether Fibonacci or Galois LFSR is implemented. Default is Fibonacci.
Gate Type	Gate type used in the feedback polynomial. The gate can be XOR or XNOR. Default is XOR.
Number of Bits	Size of LFSR that can be implemented. Number of bits ranges from 1 to 512. Default is 8.
Feedback Polynomial	Feedback polynomial that is handled by the LFSR. The polynomial limit is 64 bits. The default is 0x00.
Initial Value	Initial value of the LFSR. The initial value limit is 512 bits. Default is 0x01.
Enable Parallel Output	Enables parallel output when selected or serial output when not selected. Default is selected.

Table 205: LFSR Dialog Box (Continued)

Feature	Description
Use Reloadable Seed Values	If the check box is checked, the Din port is defined. The value on Din is a parallel value having the same number of bits as the shift register, and presents the value of the reloadable seed to the block. The reloadable seed is loaded at initialization, and whenever a new shift sequence is started. Default is not selected.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195

MAC

Two-input multiply/accumulate (MAC) logic. The module provides a variety of multiply/accumulate organizations and is implemented as embedded sysDSP Blocks. Ports, properties, and functionality are described in detail in the “See Also” section below.

The options vary with the device family. See:

- “MAC Options (ECP5, LatticeECP3)” on page 1480
- “MAC Options (LatticeECP, LatticeECP2/M, LatticeXP2)” on page 1481

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*.

MAC Options (ECP5, LatticeECP3)

Table 206: MAC Dialog Box for ECP5 and LatticeECP3

Feature	Description
Size of the DSPMAC Block	**Width** – Specify the size of the inputs. **Data Type** – Specify the data format of the inputs as signed, unsigned, or dynamic selection of the sign. **Source** – Specify if the source of the input is Parallel for generic routing, Shift for the shift outputs of an adjacent DSP block, or Dynamic for dynamic selection of the source of the input. **Accumulator** – Size of the output. This can be a number between product size + 16 and product size + 32. The product size is the sum of the multiplicand and multiplier bit sizes. The default size (product + 16) will use the accumulator in the DSP block (if possible). If the accumulator size is greater than 52, it will be implemented in PFUs. **Select Shift Out A** – Enables the shift-out port A of the DSP block. **Select Shift Out B** – Enables the shift-out port B of the DSP block. **Add/Sub Operation** – Specify addition or subtraction in the accumulator, or dynamic selection of the operation. In dynamic mode high = add, low = subtract.
Register Options	Enable or disable input, pipeline, and output registers. Then specify options for the registers. Choices for Clock, CE, and RST vary depending on which registers are enabled and whether or not a clock is selected. With some configurations, there is only one choice and it is selected automatically. **Reset Mode** – Specify whether the reset is synchronous or asynchronous. **Clock** – Selects the clock for the registers of the DSP block. If both input registers have a clock specified, the pipeline register must use the same clock as one of the input registers (InputA or InputB) or no clock (NONE). If the pipeline register has a clock specified, the output register will use that same clock (Pipeline). **CE** – Selects the clock enable pin to the registers of the DSP block. **RST** – Selects the reset pin to the registers of the DSP block. If both input registers have a clock specified, the pipeline and output resets automatically use the same sources as their clocks (InputA, InputB, or Pipeline).
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172

- TN1267, *ECP5 sysDSP Usage Guide*

MAC Options (LatticeECP, LatticeECP2/M, LatticeXP2)

The module dialog box has two tabs:

- Basic, below
- Advanced, see Table 208 on page 1483

In the Basic tab, you can enable pipelining and set parameters such as the size and format of the data. In the Advanced tab, you can control each group of pipelining registers with independent clocks, clock enables, and resets.

Table 207: MAC Dialog Box for LatticeECP, LatticeECP2/M, and LatticeXP2—Basic Tab

Feature	Description
Size of the DSPMAC block	**Multiplicand A bit size** – Specify the size of the first term in the multiplication. **Multiplier B bit size** – Specify the size of the second term in the multiplication. **Accumulator bit size** – Size of the output. This can be a number between product size + 16 and product size + 32. The product size is the sum of the multiplicand and multiplier bit sizes. The default size (product + 16) will use the accumulator in the DSP block (if possible). If the accumulator size is greater than 52, it will be implemented in PFUs.
Select Block Options	**Area/Speed** – Specify if the implementation should minimize the number of DSP blocks (Area) or minimize the number of partial products to be added (Speed). In some cases, both the options will produce the same implementation. **Accumulator Implementation** – Specify if the accumulator is built using the built-in accumulator in the sysDSP block or using PFU logic. It is possible to use the built-in accumulator when using default output size and the size is less than 52. However, only two MACs fit in a DSP block. When using PFU logic, it is possible to have four MACs per DSP block, but it will use additional logic resources. **Add/Sub Operation** – Select if the Accumulator is in add, subtract, or dynamic mode. In dynamic mode high = add, low = subtract.
Data Format	**Note**: The shift options, input and out, are only available if both inputs are 18 or fewer bits. **Input A, B** – Specify if the source of input is from generic routing (Parallel) or the shift outputs of an adjacent DSP block (Shift). The Shift format enables a sample/shift register. Dynamic control is also allowed (not available in LatticeECP). **Signed/Unsigned** – Specify if the data format of the inputs is signed or unsigned. Dynamic selection using an additional input is also available (not available in LatticeECP). **Select Shift Out A, B** – Enables the shift out on port A or port B of the DSP block.

Table 207: MAC Dialog Box for LatticeECP, LatticeECP2/M, and LatticeXP2—Basic Tab (Continued)

Feature	Description
Select Pipelining	**Enable Input Registers** – Enables the input registers. This option is automatically selected if either of the inputs has a Shift or Dynamic data format specified. See above. **Enable Pipeline Registers** – Enables the pipeline registers. **Enable Output Register** – For the MAC, the output register is always enabled. This is used to implement the accumulator. **RST on all registers** – Connects a reset pin to all registers of the DSP block. **CE on all registers** – Connects a clock enable pin to all registers of the DSP block.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 208: MAC Dialog Box for LatticeECP, LatticeECP2/M, and LatticeXP2—Advanced Tab

Feature	Description
Select Pipelining	**Enable Input Register A, B** – Specify options for Input A or Input B registers. This option is disabled if the Enable Input Registers option on the Basic tab is not selected. **Enable Signed Register A, B –** (not available in LatticeECP) Specify options for signed registers A or B of the DSP block. **Enable Signed Pipeline Register A, B –** (not available in LatticeECP) Specify options for signed pipeline registers A or B of the DSP block. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Accumulator Load Input Register** – Specify options for the accumulator load input register. This option is disabled if the Enable Input Registers option on the Basic tab is not selected. **Enable Accumulator Load Pipeline Register** – Specify options for the accumulator load pipeline register. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Addsub Input Register** – Specify options for the ADDNSUB input register. This option is enabled only if Add/Sub Operation on the Basic tab is set to Dynamic mode. **Enable Addsub Pipeline Register** – Specify options for the ADDNSUB pipeline register. This option is enabled only if Add/Sub Operation on the Basic tab is set to Dynamic mode and the Enable Pipeline Registers option is selected. **Enable Pipeline Register** – Specify options for the pipeline registers of the multiplier. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Output Register** – Specify options for the output accumulator registers. If the accumulator is implemented in the PFU, this option is used to implement the PFU registers.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*

pmi_dsp_mac

The following table is intended to provide information for pmi_dsp_mac usage (pmi_dsp_mac.v).

pmi_dsp_mac Control and Data Signal Descriptions

RST Asynchronous reset of selected registers		
SignA Dynamic signal:	0 = unsigned,	1 = signed
SignB Dynamic signal:	0 = unsigned	1 = signed
ACCUMSLOAD Dynamic signal:	0 = accumulate	1 = load
ADDNSUB Dynamic signal:	0 = subtract	1 = add
SOURCEA Dynamic signal:	0 = parallel input	1 = shift input
SOURCEB Dynamic signal:	0 = parallel input	1 = shift input

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_dsp_mac.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	CLK0	Bit	N/A
I	CLK1	Bit	N/A
I	CLK2	Bit	N/A
I	CLK3	Bit	N/A
I	CE0	Bit	N/A
I	CE1	Bit	N/A
I	CE2	Bit	N/A
I	CE3	Bit	N/A
I	RST0	Bit	N/A
I	RST1	Bit	N/A
I	RST2	Bit	N/A
I	RST3	Bit	N/A
I	SignA	Bit	N/A
I	SignB	Bit	N/A
I	SourceA	Bit	N/A
I	SourceB	Bit	N/A
I	ADDNSUB	Bit	N/A
I	A	Bus	(pmi_dataa_width -1):0
I	B	Bus	(pmi_datab_width -1):0
I	LD	Bus	((pmi_dataa_width + pmi_datab_width - 1) + 16):0
I	ACCUMSLOAD	Bit	N/A
I	SRIA	Bus	17:0
I	SRIB	Bus	17:0
O	OVERFLOW	Bit	N/A
O	ACCUM	Bus	((pmi_dataa_width + pmi_datab_width - 1) + 16):0
O	SROA	Bus	17:0
O	SROB	Bus	17:0

Verilog pmi_dsp_mac Definition

```
module pmi_dsp_mac #(
parameter pmi_dataa_width = 8,
   parameter pmi_datab_width = 8,
```

```
 parameter pmi_additional_pipeline = 0,
 parameter pmi_input_reg = "on",
 parameter pmi_family = "ECP2",
 parameter pmi_gsr = "disable",
 parameter pmi_source_control_a = "parallel",
 parameter pmi_source_control_b = "parallel",
 parameter pmi_reg_inputa_clk = "CLK0",
 parameter pmi_reg_inputa_ce = "CE0",
 parameter pmi_reg_inputa_rst = "RST0",
 parameter pmi_reg_inputb_clk = "CLK0",
 parameter pmi_reg_inputb_ce = "CE0",
 parameter pmi_reg_inputb_rst = "RST0",
 parameter pmi_reg_pipeline_clk = "CLK0",
 parameter pmi_reg_pipeline_ce = "CE0",
 parameter pmi_reg_pipeline_rst = "RST0",
 parameter pmi_reg_output_clk = "CLK0",
 parameter pmi_reg_output_ce = "CE0",
 parameter pmi_reg_output_rst = "RST0",
 parameter pmi_reg_signeda_0_clk = "CLK0",
 parameter pmi_reg_signeda_0_ce = "CE0",
 parameter pmi_reg_signeda_0_rst = "RST0",
 parameter pmi_reg_signeda_1_clk = "CLK0",
 parameter pmi_reg_signeda_1_ce = "CE0",
 parameter pmi_reg_signeda_1_rst = "RST0",
 parameter pmi_reg_signedb_0_clk = "CLK0",
 parameter pmi_reg_signedb_0_ce = "CE0",
 parameter pmi_reg_signedb_0_rst = "RST0",
 parameter pmi_reg_signedb_1_clk = "CLK0",
 parameter pmi_reg_signedb_1_ce = "CE0",
 parameter pmi_reg_signedb_1_rst = "RST0",
 parameter pmi_reg_addnsub_0_clk = "CLK0",
 parameter pmi_reg_addnsub_0_ce = "CE0",
 parameter pmi_reg_addnsub_0_rst = "RST0",
 parameter pmi_reg_addnsub_1_clk = "CLK0",
 parameter pmi_reg_addnsub_1_ce = "CE0",
 parameter pmi_reg_addnsub_1_rst = "RST0",
 parameter pmi_reg_accumsload_0_clk = "CLK0",
 parameter pmi_reg_accumsload_0_ce = "CE0",
 parameter pmi_reg_accumsload_0_rst = "RST0",
 parameter pmi_reg_accumsload_1_clk = "CLK0",
 parameter pmi_reg_accumsload_1_ce = "CE0",
 parameter pmi_reg_accumsload_1_rst = "RST0",
 parameter module_type = "pmi_dsp_mac")

(input [(pmi_dataa_width-1):0]       A,
 input [(pmi_datab_width-1):0]       B,
 input [17:0]       SRIA,
 input [17:0]       SRIB,
 input CLK0,
 input CLK1,
 input CLK2,
 input CLK3,
 input CE0,
 input CE1,
 input CE2,
 input CE3,
 input RST0,
 input RST1,
 input RST2,
```

```
   input RST3,
   input SignA,
   input SignB,
   input SourceA,
   input SourceB,
   input ADDNSUB,
   input [((pmi_dataa_width + pmi_datab_width - 1) + 16):0] LD,
   input ACCUMSLOAD,
   output [((pmi_dataa_width + pmi_datab_width - 1) + 16):0]
ACCUM,
   output OVERFLOW,
   output [17:0]  SROA,
   output [17:0]  SROB)/*synthesis syn_black_box */;
endmodule // pmi_dsp_mac
```

VHDL pmi_dsp_mac Definition

```
component pmi_dsp_mac is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       pmi_additional_pipeline : integer := 1;
       pmi_input_reg : string := "on";
       pmi_family : string := "ECP2";
       pmi_gsr : string := "disable";
       pmi_source_control_a : string := "parallel";
       pmi_source_control_b : string := "parallel";
       pmi_reg_inputa_clk : string := "CLK0";
       pmi_reg_inputa_ce : string := "CE0";
       pmi_reg_inputa_rst : string := "RST0";
       pmi_reg_inputb_clk : string := "CLK0";
       pmi_reg_inputb_ce : string := "CE0";
       pmi_reg_inputb_rst : string := "RST0";
       pmi_reg_pipeline_clk : string := "CLK0";
       pmi_reg_pipeline_ce : string := "CE0";
       pmi_reg_pipeline_rst : string := "RST0";
       pmi_reg_output_clk : string := "CLK0";
       pmi_reg_output_ce : string := "CE0";
       pmi_reg_output_rst : string := "RST0";
       pmi_reg_signeda_0_clk : string := "CLK0";
       pmi_reg_signeda_0_ce : string := "CE0";
       pmi_reg_signeda_0_rst : string := "RST0";
       pmi_reg_signeda_1_clk : string := "CLK0";
       pmi_reg_signeda_1_ce : string := "CE0";
       pmi_reg_signeda_1_rst : string := "RST0";
       pmi_reg_signedb_0_clk : string := "CLK0";
       pmi_reg_signedb_0_ce : string := "CE0";
       pmi_reg_signedb_0_rst : string := "RST0";
       pmi_reg_signedb_1_clk : string := "CLK0";
       pmi_reg_signedb_1_ce : string := "CE0";
       pmi_reg_signedb_1_rst : string := "RST0";
       pmi_reg_addnsub_0_clk : string := "CLK0";
       pmi_reg_addnsub_0_ce : string := "CE0";
       pmi_reg_addnsub_0_rst : string := "RST0";
       pmi_reg_addnsub_1_clk : string := "CLK0";
       pmi_reg_addnsub_1_ce : string := "CE0";
       pmi_reg_addnsub_1_rst : string := "RST0";
       pmi_reg_accumsload_0_clk : string := "CLK0";
       pmi_reg_accumsload_0_ce : string := "CE0";
```

```
        pmi_reg_accumsload_0_rst : string := "RST0";
        pmi_reg_accumsload_1_clk : string := "CLK0";
        pmi_reg_accumsload_1_ce : string := "CE0";
        pmi_reg_accumsload_1_rst : string := "RST0";
        module_type : string := "pmi_dsp_mac"
    );
    port (
     A:                      in
std_logic_vector((pmi_dataa_width-1) downto 0);
     B:                      in
std_logic_vector((pmi_datab_width-1) downto 0);
     SRIA:                   in std_logic_vector(17 downto 0);
     SRIB:                   in std_logic_vector(17 downto 0);
     CLK0, CLK1, CLK2, CLK3: in std_logic;
     CE0, CE1, CE2, CE3:     in std_logic;
     RST0, RST1, RST2, RST3: in std_logic;
     SignA:                  in std_logic;
     SignB:                  in std_logic;
     SourceA:                in std_logic;
     SourceB:                in std_logic;
     ADDNSUB:                in std_logic;
     LD:                     in
std_logic_vector(((pmi_dataa_width + pmi_datab_width - 1) + 16)
downto 0);
     ACCUMSLOAD:             in std_logic;
     ACCUM:                  out
std_logic_vector(((pmi_dataa_width + pmi_datab_width - 1) + 16)
downto 0);
     OVERFLOW:               out std_logic;
     SROA:                   out std_logic_vector(17 downto 0);
     SROB:                   out std_logic_vector(17 downto 0)
   );
  end component pmi_dsp_mac;
```

Parameter Names and Values

The following table contains the pmi_dsp_mac module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 72	8	
pmi_datab_width	2 to 72	8	
pmi_gsr	“enable” | “disable”	“enable”	
pmi_additional_pipeline	0 to 1	0	
pmi_input_reg	“on” | “off”	“on”	
pmi_family	“ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “XP2”	“ECP2”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_source_controla	“parallel”	“shift”	“parallel”	The pmi_source_controla and pmi_source_controlb parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controlb	“parallel”	“shift”	“parallel”	The pmi_source_controla and pmi_source_controlb parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_reg_inputa_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_inputa_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_inputa_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_inputb_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_inputb_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_inputb_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_pipeline_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_pipeline_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_pipeline_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_output_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_output_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_output_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signeda_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signeda_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signeda_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signeda_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signeda_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signeda_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_reg_signedb_0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_signedb_0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_signedb_0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_signedb_1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_signedb_1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_signedb_1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_accumsload_0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_accumsload_0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_accumsload_0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_accumsload_1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_accumsload_1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_accumsload_1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_addnsub_0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_addnsub_0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_addnsub_0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_addnsub_1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_addnsub_1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_addnsub_1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

MIPI_DPHY

MIPI DPHY uses a point-to-point differential interface, with modular architecture that supports multiple data lanes and a clock lane, allowing all possible configurations.

Table 209: MIPI_DPHY Dialog Box

Feature	Description
Interface Type	Select the Transmit or Receive interface type.
MIPI Interface Application	Select a DSI or CSI-2 application.
Receiver DPHY Module Type	In Hard MIPI DPHY mode, Scuba will generate the MIPI Hard DPHY Rx and Tx interface modules. In Soft MIPI DPHY mode, Scuba will generate the Soft MIPI DPHY Rx interface module.
Interface Clock Frequency	This sets the Frequency MIPI Input Clock, which determines the gearing ratio used for DPHY Soft and Hard modules.
Interface Data Rate	A read-only field displaying a calculated value based on the clock frequency.
Gearing Ratio	This value is set automatically based on the clock frequency. It can also be manually adjusted.
Bus Width	The desired bus width to generate.
DPHY PLL Input Reference Clock Frequency	Input Clock Frequency of the DPHY PLL. Based on this value and the divider settings, Scuba automatically calculates the output clock frequency. The CLKOUT frequency will be the interface Clock Frequency entered above.
Reference Clock from I/O Pin	Used to set the IO_TYPE for the reference clock input to the PLL so that the Planner can place the module.
Reference Clock Input Buffer Type	Used to set the IO_TYPE for the reference clock input to the PLL so that the Planner can place the module.
Include Start up Synchronization logic	Adds start-up synchronization logic.
Configure	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195

MMAC

Multiplier Multiplier Accumulate provides a variety of multiply/accumulate organizations and is implemented as embedded sysDSP Blocks.

Table 210: MMAC Dialog Box

Feature	Description
Size of the DSPMMAC Block	**Width** – Specify the size of the inputs. **Data Type** – Specify the data format of the inputs as signed, unsigned, or dynamic selection of the sign. **Source** – Specify if the source of the input is Parallel for generic routing, Shift for the shift outputs of an adjacent DSP block, or Dynamic for dynamic selection of the source of the input. **Accumulator** – Size of the output. This is Input A0/A1 + Input B0/B1 + 16. The output uses the accumulator in the DSP block if possible. If the accumulator size is greater than 52, it is implemented in PFUs. **Select Shift Out A** – Enables the shift-out port A of the DSP block. **Select Shift Out B** – Enables the shift-out port B of the DSP block. **Operation** – Specify addition or subtraction of the two multiplication results, or dynamic selection of the operation. In dynamic mode high = add, low = subtract.
Register Options	**Reset Mode** – Specify whether the reset is synchronous or asynchronous. **Clock** – Selects the clock for the registers of the DSP block. **CE** – Selects the clock enable pin to the registers of the DSP block. **RST** – Selects the reset pin to the registers of the DSP block.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172

MULT

Multiplier that provides a variety of multiplier organizations and is implemented as embedded sysDSP Blocks. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

The options vary with the device family. See:

- “MULT Options (ECP5, LatticeECP3)” on page 1493
- “MULT Options (LatticeECP, LatticeECP2/M, LatticeXP2)” on page 1494

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*

MULT Options (ECP5, LatticeECP3)

Table 211: MULT Dialog Box for ECP5 and LatticeECP3

Feature	Description
Size of the DSPMULT Block	**Width** – Specify the size of the inputs. **Data Type** – Specify the data format of the inputs as signed, unsigned, or dynamic selection of the sign. **Source** – Specify if the source of the input is Parallel for generic routing, Shift for the shift outputs of an adjacent DSP block, or Dynamic for dynamic selection of the source of the input. **Output** – Size of the output. The product size is the sum of the multiplicand and multiplier bit sizes. **Select Shift Out A** – Enables the shift-out port A of the DSP block. **Select Shift Out B** – Enables the shift-out port B of the DSP block.
Register Options	**Reset Mode** – Specify whether the reset is synchronous or asynchronous. **Clock** – Selects the clock for the registers of the DSP block. **CE** – Selects the clock enable pin to the registers of the DSP block. **RST** – Selects the reset pin to the registers of the DSP block.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysDSP" on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*

MULT Options (LatticeECP, LatticeECP2/M, LatticeXP2)

In the Basic tab you set most of the parameters including the size and format of the data, and enable pipelining. In the Advanced tab, you can control each group of pipelining registers with independent clocks, clock enables, and resets.

The module dialog box has two tabs:

- Basic, below
- Advanced, see Table 213 on page 1495

Table 212: MULT Dialog Box LatticeECP, LatticeECP2/M, and LatticeXP2—Basic Tab

Feature	Description
Size of the DSPMULT Block	**Multiplicand A bit size** – Specify the size of the first term in the multiplication. **Multiplier B bit size** – Specify the size of the second term in the multiplication. **Product bit size** – Size of the output. This is an automatically calculated value. The product size is the sum of the multiplicand and multiplier bit sizes.
Select Block Options	**Area/Speed** – Specify if the implementation should minimize the number of DSP blocks (Area) or minimize the number of partial products to be added (Speed). In some cases, both the options will produce the same implementation.
Data Format	**Note**: The shift options, input and out, are not available if either of the A and B inputs is larger than 18 bits. **Input A, B** – Specify if the source of input is from generic routing (Parallel) or the shift outputs of an adjacent DSP block (Shift). The Shift format enables a sample/shift register. Dynamic control is also allowed (not available in LatticeECP). **Signed/Unsigned** – Specify if the data format of the inputs is signed or unsigned. Dynamic selection of signed/unsigned using an additional input is also allowed (not available in LatticeECP). **Select Shift Out A, B** – Enables the shift-out on port A or port B of the DSP block.

Table 212: MULT Dialog Box LatticeECP, LatticeECP2/M, and LatticeXP2—Basic Tab (Continued)

Feature	Description
Select Pipelining	**Enable Input Registers** – Enables the input registers. This option is automatically selected if either of the inputs has a Shift or Dynamic data format specified. See above. **Enable Pipeline Registers** – Enables the pipeline registers. **Enable Output Register** – Enables the output register. **RST on all registers** – Connects a reset pin to all registers of the DSP block. **CE on all registers** – Connects a clock enable pin to all registers of the DSP block.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 213: MULT Dialog Box LatticeECP, LatticeECP2/M, and LatticeXP2—Advanced Tab

Feature	Description
Select Pipelining	**Enable Input Register A, B** – Specify options for Input A or Input B registers. This option is disabled if the Enable Input Registers option on the Basic tab is not selected. **Enable Signed Register A, B –** (not available in LatticeECP) Specify options for signed registers A or B of the DSP block. This option is disabled unless the Signed/ Unsigned A, B option on the Basic tab is set to Dynamic and the Enable Input Registers option is selected. **Enable Signed Pipeline Register A, B –** (not available in LatticeECP) Specify options for signed pipeline registers A or B of the DSP block. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Pipeline Register** – Specify options for the pipeline registers of the multiplier. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Output Register** – Specify options for the output accumulator registers. If the accumulator is implemented in the PFU, this option is used to implement the PFU registers. This option is disabled if the Enable Output Register option on the Basic tab is not selected.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysDSP" on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*

pmi_dsp_mult

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_dsp_mult.

Port Description

Input/ Output	Port Name	Type	Size (Buses Only)	Description
I	CLK0	Bit	N/A	Clock input.
I	CLK1	Bit	N/A	Clock input.
I	CLK2	Bit	N/A	Clock input.
I	CLK3	Bit	N/A	Clock input.
I	CE0	Bit	N/A	Clock enable input
I	CE1	Bit	N/A	Clock enable input
I	CE2	Bit	N/A	Clock enable input
I	CE3	Bit	N/A	Clock enable input
I	RST0	Bit	N/A	Reset input
I	RST1	Bit	N/A	Reset input
I	RST2	Bit	N/A	Reset input
I	RST3	Bit	N/A	Reset input
I	SignA	Bit	N/A	Signed bit input for A
I	SignB	Bit	N/A	Signed bit input for B
I	SourceA	Bit	N/A	Source Selector for Multiplier Input A
I	SourceB	Bit	N/A	Source Selector for Multiplier Input b
I	A	Bus	(pmi_dataa_width - 1):0	Multiplier parallel Input A
I	B	Bus	(pmi_datab_width - 1):0	Multiplier parallel Input b
I	SRIA	Bus	17:0	Multiplier shift Input A
I	SRIB	Bus	17:0	Multiplier shift Input B

Input/ Output	Port Name	Type	Size (Buses Only)	Description
O	P	Bus	(pmi_dataa_width + pmi_datab_width - 1):0	Product Output
O	SROA	Bus	17:0	Shift Output A
O	SROB	Bus	17:0	Shift Output B

Verilog pmi_dsp_mult Definition

```
module pmi_dsp_mult
 #(parameter pmi_dataa_width = 8,
   parameter pmi_datab_width = 8,
   parameter pmi_additional_pipeline = 0,
   parameter pmi_input_reg = "on",
   parameter pmi_output_reg = "on",
   parameter pmi_family = "ECP2",
   parameter pmi_gsr = "disable",
   parameter pmi_source_control_a = "parallel",
   parameter pmi_source_control_b = "parallel",
   parameter pmi_reg_inputa_clk = "CLK0",
   parameter pmi_reg_inputa_ce = "CE0",
   parameter pmi_reg_inputa_rst = "RST0",
   parameter pmi_reg_inputb_clk = "CLK0",
   parameter pmi_reg_inputb_ce = "CE0",
   parameter pmi_reg_inputb_rst = "RST0",
   parameter pmi_reg_pipeline_clk = "CLK0",
   parameter pmi_reg_pipeline_ce = "CE0",
   parameter pmi_reg_pipeline_rst = "RST0",
   parameter pmi_reg_output_clk = "CLK0",
   parameter pmi_reg_output_ce = "CE0",
   parameter pmi_reg_output_rst = "RST0",
   parameter pmi_reg_signeda_clk = "CLK0",
   parameter pmi_reg_signeda_ce = "CE0",
   parameter pmi_reg_signeda_rst = "RST0",
   parameter pmi_reg_signedb_clk = "CLK0",
   parameter pmi_reg_signedb_ce = "CE0",
   parameter pmi_reg_signedb_rst = "RST0",
   parameter module_type = "pmi_dsp_mult")

  (input [(pmi_dataa_width-1):0]       A,
   input [(pmi_datab_width-1):0]       B,
   input [17:0]       SRIA,
   input [17:0]       SRIB,
   input CLK0,
   input CLK1,
   input CLK2,
   input CLK3,
   input CE0,
   input CE1,
   input CE2,
   input CE3,
   input RST0,
   input RST1,
   input RST2,
   input RST3,
   input SignA,
```

```
    input SignB,
    input SourceA,
    input SourceB,
    output [(pmi_dataa_width + pmi_datab_width - 1):0]  P,
    output [17:0]  SROA,
    output [17:0]  SROB)/*synthesis syn_black_box */;

endmodule //pmi_dsp_mult
```

VHDL pmi_dsp_mult Definition

```
component pmi_dsp_mult is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       pmi_additional_pipeline : integer := 0;
       pmi_input_reg : string := "on";
       pmi_output_reg : string := "on";
       pmi_family : string := "ECP2";
       pmi_gsr : string := "disable";
       pmi_source_control_a : string := "parallel";
       pmi_source_control_b : string := "parallel";
       pmi_reg_inputa_clk : string := "CLK0";
       pmi_reg_inputa_ce : string := "CE0";
       pmi_reg_inputa_rst : string := "RST0";
       pmi_reg_inputb_clk : string := "CLK0";
       pmi_reg_inputb_ce : string := "CE0";
       pmi_reg_inputb_rst : string := "RST0";
       pmi_reg_pipeline_clk : string := "CLK0";
       pmi_reg_pipeline_ce : string := "CE0";
       pmi_reg_pipeline_rst : string := "RST0";
       pmi_reg_output_clk : string := "CLK0";
       pmi_reg_output_ce : string := "CE0";
       pmi_reg_output_rst : string := "RST0";
       pmi_reg_signeda_clk : string := "CLK0";
       pmi_reg_signeda_ce : string := "CE0";
       pmi_reg_signeda_rst : string := "RST0";
       pmi_reg_signedb_clk : string := "CLK0";
       pmi_reg_signedb_ce : string := "CE0";
       pmi_reg_signedb_rst : string := "RST0";
       module_type : string := "pmi_dsp_mult"
    );
    port (
     A : in std_logic_vector((pmi_dataa_width-1) downto 0);
     B : in std_logic_vector((pmi_datab_width-1) downto 0);
     SRIA : in std_logic_vector(17 downto 0);
     SRIB : in std_logic_vector(17 downto 0);
     CLK0: in std_logic;
     CLK1: in std_logic;
     CLK2: in std_logic;
     CLK3: in std_logic;
     CE0: in std_logic;
     CE1: in std_logic;
     CE2: in std_logic;
     CE3: in std_logic;
     RST0: in std_logic;
     RST1: in std_logic;
     RST2: in std_logic;
     RST3: in std_logic;
```

```
      SignA: in std_logic;
      SignB: in std_logic;
      SourceA: in std_logic;
      SourceB: in std_logic;
      P : out std_logic_vector((pmi_dataa_width +
pmi_datab_width - 1) downto 0);
      SROA : out std_logic_vector(17 downto 0);
      SROB : out std_logic_vector(17 downto 0)
    );
  end component pmi_dsp_mult;
```

Parameter Names and Values

The following table contains the pmi_dsp_mult module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 72	8	
pmi_datab_width	2 to 72	8	
pmi_gsr	“enable” | “disable”	“enable”	
pmi_additional_pipeline	0 to 1	0	
pmi_input_reg	“on” | “off”	“on”	
pmi_output_reg	“on” | “off”	“on”	
pmi_source_controla	“parallel” | “shift”	“parallel”	The pmi_source_controla and pmi_source_controlb parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controlb	“parallel” | “shift”	“parallel”	The pmi_source_controla and pmi_source_controlb parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_family	“ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “XP2”	“ECP2”	
reg_inputa_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
reg_inputa_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
reg_inputa_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
reg_inputb_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
reg_inputb_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
reg_inputb_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
reg_pipeline_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	

Parameter Name	Acceptable Values	Default Value	Notes
reg_pipeline_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
reg_pipeline_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
reg_output_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
reg_output_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
reg_output_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
reg_signeda_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
reg_signeda_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
reg_signeda_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
reg_signedb_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
reg_signedb_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
reg_signedb_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

Mult_Add_Sub

Two-input MULTADDSUB block look-up table (LUT) implementation. For DSP devices, DSP implementation is also available.

Table 214: Mult_Add_Sub Dialog Box

Feature	Description
Block Implementation	Allows you to choose between LUT-based implementation and DSP. DSP is only available with ECP5, LatticeECP, LatticeECP2/M, LatticeECP3, and LatticeXP2.
Add/Sub Operation	Allows the implementation of adder or subtractor. Default is set to add.
Input A0/A1 Width	Size of multiplier inputs A0/A1. The size of the multiplier inputs can be any size between 2 and 36 bits. The default size is 9.
Input B0/B1 Width	Size of multiplicand inputs B0/B1. The size of the multiplicand inputs can be any size between 2 and 36 bits. The default size is 9.
Representation	Allows the operation of signed and unsigned multiplication. Default setting is signed.

Table 214: Mult_Add_Sub Dialog Box (Continued)

Feature	Description
Sum Width	Sum of the multiplier and multiplicand inputs + 1.
Specify the Number of Pipeline Stages	Allows the number of pipelined stages to be set. Default is set to 1.
Enable Input Registers	Allows the implementation of registers on the inputs. Default is input register is enabled.
Enable Output Registers	Allows the implementation of registers on the outputs. Default is output register is enabled.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195

pmi_multaddsub

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_multaddsub.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA0	Bus	(pmi_dataa_width -1):0
I	DataA1	Bus	(pmi_dataa_width -1):0
I	DataB0	Bus	(pmi_datab_width -1):0
I	DataB1	Bus	(pmi_datab_width -1):0
I	Clock	Bit	N/A
I	ClkEn	Bit	N/A
I	Aclr	Bit	N/A
O	Result	Bus	(pmi_dataa_width + pmi_datab_width):0

Verilog pmi_multaddsub Definition

```
module pmi_multaddsub #(parameter pmi_dataa_width = 8,
         parameter pmi_datab_width = 8,
         parameter module_type = "pmi_multaddsub",
         parameter pmi_sign = "on",
         parameter pmi_additional_pipeline = 1,
         parameter pmi_add_sub = "add",
         parameter pmi_input_reg = "on",
         parameter pmi_output_reg = "on",
         parameter pmi_family = "EC",
         parameter pmi_implementation = "LUT")

        (input   [(pmi_dataa_width-1):0]  DataA0,
         input   [(pmi_dataa_width-1):0]  DataA1,
         input   [(pmi_datab_width-1):0]  DataB0,
         input   [(pmi_datab_width-1):0]  DataB1,
         input   Clock, ClkEn, Aclr,
         output  [(pmi_dataa_width + pmi_datab_width):0]
Result)/*synthesis syn_black_box*/;

endmodule // pmi_multaddsub
```

VHDL pmi_multaddsub Definition

```
component pmi_multaddsub is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       module_type : string := "pmi_multaddsub";
       pmi_sign : string := "on";
       pmi_additional_pipeline : integer := 1;
       pmi_add_sub : string := "add";
       pmi_input_reg : string := "on";
       pmi_output_reg : string := "on";
       pmi_family : string := "EC";
       pmi_implementation : string := "LUT"
    );
    port (
```

```
     DataA0 : in std_logic_vector((pmi_dataa_width-1) downto
0);
     DataA1 : in std_logic_vector((pmi_dataa_width-1) downto
0);
     DataB0 : in std_logic_vector((pmi_datab_width-1) downto
0);
     DataB1 : in std_logic_vector((pmi_datab_width-1) downto
0);
     Clock: in std_logic;
     Result : out std_logic_vector((pmi_dataa_width +
pmi_datab_width) downto 0)
   );
  end component pmi_multaddsub;
```

Parameter Names and Values

The following table contains the pmi_multaddsub module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 36	8	
pmi_datab_width	2 to 36	8	
pmi_sign	“on” | “off”	“on”	
pmi_additional_pipeline	0 to Maximum {pmi_dataa_width, pmi_datab_width}	1	The pmi_additional_pipeline parameter ranges from 0 to 1 for DSP implementation. For the LUT implementation the limit is set to Maximum {pmi_dataa_width, pmi_datab_width}.
pmi_add_sub	“add” | “sub”	“add”	
pmi_input_reg	“on” | “off”	“on”	
pmi_output_reg	“on” | “off”	“on”	
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” |“LPTM”	“EC”	
pmi_implementation	“DSP” | “LUT”	“LUT”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

Mult_Add_Sub_Sum

Two-input MULTADDSUBSUM block look-up table (LUT) implementation. For DSP devices, DSP implementation is also available.

Table 215: Mult_Add_Sub_Sum Dialog Box

Feature	Description
Block Implementation	Allows you to choose between LUT-based implementation and DSP. DSP is only available with LatticeECP, LatticeECP2/M, LatticeECP3, ECP5, and LatticeXP2.
Add/Sub 0 Operation	Allows the implementation of adder or subtractor. Default is set to add.
Add/Sub 1 Operation	Allows the implementation of adder or subtractor. Default is set to add.
Input A0/A1/A3/A3 Width	Size of multiplier inputs A0/A1/A2/A3. The size of the multiplier inputs can be any size between 2 and 36 bits. The default size is 9.
Input B0/B1/B2/B3 Width	Size of multiplicand inputs B0/B1/B2/B3. The size of the multiplicand inputs can be any size between 2 and 36 bits. The default size is 9.
Representation	Allows the operation of signed and unsigned multiplication. Default setting is signed.
Product Bit Width	Sum of the multiplier and multiplicand inputs + 2.
Specify the Number of Pipeline Stages	Allows the number of pipelined stages to be set. Default is set to 1.
Enable Input Registers	Allows the implementation of registers on the inputs. Default is input register is enabled.
Enable Output Registers	Allows the implementation of registers on the outputs. Default is output register is enabled.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195

pmi_multaddsubsum

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_multaddsubsum.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA0	Bus	(pmi_dataa_width -1):0
I	DataA1	Bus	(pmi_dataa_width -1):0
I	DataA2	Bus	(pmi_dataa_width -1):0
I	DataA3	Bus	(pmi_dataa_width -1):0
I	DataB0	Bus	(pmi_datab_width -1):0
I	DataB1	Bus	(pmi_datab_width -1):0
I	DataB2	Bus	(pmi_datab_width -1):0
I	DataB3	Bus	(pmi_datab_width -1):0
I	Clock	Bit	N/A
I	ClkEn	Bit	N/A
I	Aclr	Bit	N/A
O	Result	Bus	(pmi_dataa_width + pmi_datab_width + 1):0

Verilog pmi_multaddsubsum Definition

```
module pmi_multaddsubsum #(parameter pmi_dataa_width = 8,
         parameter pmi_datab_width = 8,
         parameter module_type = "pmi_multaddsubsum",
         parameter pmi_sign = "on",
         parameter pmi_additional_pipeline = 1,
         parameter pmi_add_sub0 = "add",
         parameter pmi_add_sub1 = "add",
         parameter pmi_input_reg = "on",
         parameter pmi_output_reg = "on",
         parameter pmi_family = "EC",
         parameter pmi_implementation = "LUT")

        (input   [(pmi_dataa_width-1):0]  DataA0,
         input   [(pmi_dataa_width-1):0]  DataA1,
         input   [(pmi_dataa_width-1):0]  DataA2,
         input   [(pmi_dataa_width-1):0]  DataA3,
         input   [(pmi_datab_width-1):0]  DataB0,
         input   [(pmi_datab_width-1):0]  DataB1,
         input   [(pmi_datab_width-1):0]  DataB2,
         input   [(pmi_datab_width-1):0]  DataB3,
         input   Clock, ClkEn, Aclr,
         output  [(pmi_dataa_width + pmi_datab_width + 1):0]
Result)/*synthesis syn_black_box*/;

endmodule // pmi_multaddsubsum
```

VHDL pmi_multaddsubsum Definition

```
component pmi_multaddsubsum is
     generic (
```

```
            pmi_dataa_width : integer := 8;
            pmi_datab_width : integer := 8;
            module_type : string := "pmi_multaddsubsum";
            pmi_sign : string := "on";
            pmi_additional_pipeline : integer := 1;
            pmi_add_sub0 : string := "add";
            pmi_add_sub1 : string := "add";
            pmi_input_reg : string := "on";
            pmi_output_reg : string := "on";
            pmi_family : string := "EC";
            pmi_implementation : string := "LUT"
        );
        port (
         DataA0 : in std_logic_vector((pmi_dataa_width-1) downto
0);
         DataA1 : in std_logic_vector((pmi_dataa_width-1) downto
0);
         DataA2 : in std_logic_vector((pmi_dataa_width-1) downto
0);
         DataA3 : in std_logic_vector((pmi_dataa_width-1) downto
0);
         DataB0 : in std_logic_vector((pmi_datab_width-1) downto
0);
         DataB1 : in std_logic_vector((pmi_datab_width-1) downto
0);
         DataB2 : in std_logic_vector((pmi_datab_width-1) downto
0);
         DataB3 : in std_logic_vector((pmi_datab_width-1) downto
0);
         Clock: in std_logic;
         Result : out std_logic_vector((pmi_dataa_width +
pmi_datab_width + 1) downto 0)
       );
      end component pmi_multaddsubsum;
```

Parameter Names and Values

The following table contains the pmi_multaddsubsum module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 36	8	
pmi_datab_width	2 to 36	8	
pmi_sign	“on” | “off”	“on”	
pmi_additional_pipeline	0 to Maximum {pmi_dataa_width, pmi_datab_width}	1	The pmi_additional_pipeline parameter ranges from 0 to 1 for DSP implementation. For the LUT implementation the limit is set to Maximum {pmi_dataa_width, pmi_datab_width}.
pmi_add_sub0	“add” | “sub”	“add”	
pmi_add_sub1	“add” | “sub”	“add”	
pmi_input_reg	“on” | “off”	“on”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_output_reg	“on”	“off”	“on”	
pmi_family	“EC”	“ECP”	“ECP2”	“ECP2M”	“ECP3”	“ECP5U”	“ECP5UM”	“SC”	“SCM”	“XP”	“XP2”	“XO”	“XO2”	“XO3L” |“LPTM”	“EC”	
pmi_implementation	“DSP”	“LUT”	“LUT”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

MULTADDSUB

Two-input MULTADDSUB block multiplier that provides a variety of multiplier add/subtract organizations and is implemented as embedded sysDSP Blocks. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

The options vary with the device family. See:

- “MULTADDSUB Options (ECP5, LatticeECP3)” on page 1508
- “MULTADDSUB Options (LatticeECP, LatticeECP2/M, LatticeXP2)” on page 1509

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*

MULTADDSUB Options (ECP5, LatticeECP3)

Table 216: MULTADDSUB Dialog Box for ECP5 and LatticeECP3

Feature	Description
Size of the DSPMULTADDSUB block	**Enable Cascade Input** – Select if this module will also get input from another MULTADDSUB module. Enabling cascade input reduces the number of bits available for the multiplier input buses (below). The output width is always 54 bits. This is needed to feed into the 54-bit CIN of the next slice. For the final slice of the chain, some lower bits may have to be ignored. Due to the alignment of the inputs to the primitives, the lower (36 - (A0/A1 width + B0/B1 width)) must be ignored. If both the input widths are less than or equal to 9, then the lower (18 - (A0/A1 width + B0/B1 width)) must be ignored. For example, if Input A0/A1 width = 12 and Input B0/B1 width = 16, the lower 36 - (12+16) = 8 bits should be ignored. **Input A0/A1 width** – Specify the size of the first term in the multiplication (for both multipliers). **Input B0/B1 width** – Specify the size of the second term in the multiplication (for both multipliers). **Sum output width** – Size of the output. This is one more than the product size (that is, it is equal to A0/A1 size + B0/B1 size + 1).
Select Options	**Select Shift Out A** - Enables the shift-out port A of the DSP block. **Cascade Match Register** – Enables the Cascade Match Register of the A multiplier. **Select Shift Out B** - Enables the shift-out port B of the DSP block. **Add/Sub Operation** - Select if the adder is adding, subtracting or dynamic mode. In dynamic mode high = add, low = subtract.
Data Options	**Input A0, A1, B0, B1 Source** – Specify if the source of the input is Parallel for generic routing, Shift for the shift outputs of an adjacent DSP block, or Dynamic for dynamic selection of the source of the input. **Data Type A, B** – Specify the data format of the inputs as signed, unsigned, or dynamic selection of the sign.
Register Options	**Reset Mode** – Specify whether the reset is synchronous or asynchronous. **Clock** – Selects the clock for the registers of the DSP block. **CE** – Selects the clock enable pin to the registers of the DSP block. **RST** – Selects the reset pin to the registers of the DSP block.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysDSP" on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*

MULTADDSUB Options (LatticeECP, LatticeECP2/M, LatticeXP2)

The module dialog box has two tabs:

- Basic, below
- Advanced, see Table 218 on page 1511

In the Basic tab you set most of the parameters including the size and format of the data, and enable pipelining. In the Advanced tab, you can control each group of pipelining registers with independent clocks, clock enables, and resets.

Table 217: MULTADDSUB Dialog Box for LatticeECP, LatticeECP2/M, and LatticeXP2—Basic Tab

Feature	Description
Size of the DSPMADD Block	**Multiplicand A0/A1 bit size** – Specify the size of the first term in the multiplication (for both the multipliers). **Multiplier B0/B1 bit size** – Specify the size of the second term in the multiplication (for both multipliers). **Sum output bit size** – Size of the output. This is one more than the product size (that is, it is equal to A0/A1 size + B0/B1 size + 1).
Select Block Options	**Area/Speed** – Specify if the implementation should minimize the number of DSP blocks (Area) or minimize number of partial products to be added (Speed). In some cases, both the options will produce the same implementation. **Add/Sub Implementation** - (not available in LatticeECP) Specify if the ADDER is built using the add/sub block in the DSP block or using PFU logic. **Add/Sub Operation** - Select if the ADDER is in add, subtract, or dynamic mode. In dynamic mode high = add, low = subtract.

Table 217: MULTADDSUB Dialog Box for LatticeECP, LatticeECP2/M, and LatticeXP2—Basic Tab

Feature	Description
Data Format	**Note**: The shift options, input and out, are not available if either of the A and B inputs is larger than 18 bits. **Input A0** – Specify if the source of input A for multiplier 0 is input A0 from generic routing (Parallel) or SRIA, the shift outputs of adjacent DSP block (Shift). Dynamic selection of the input source is also allowed (not available in LatticeECP). **Input A1** – Specify if the source of input A for multiplier 1 is input A1 from generic routing (Parallel) or the registered input A0 (Shift). Dynamic selection of the input source is also allowed (not available in LatticeECP). **Select Shift Out A** - Enables the shift-out port A of the DSP block. **Input B0** – Specify if the source of input B for multiplier 0 is input B0 from generic routing (Parallel) or SRIB, the shift outputs of the adjacent DSP block (Shift). Dynamic selection of the input source is also allowed (not available in LatticeECP). **Input B1** – Specify if the source of input B for multiplier 1 is input B1 from generic routing (Parallel) or the registered input B0 (Shift). Dynamic selection of the input source is also allowed (not available in LatticeECP). **Select Shift Out B** - Enables the shift-out port B of the DSP block. **Signed/Unsigned** – Specify if the data format of the inputs is signed or unsigned. Dynamic selection of the sign is also allowed (not available in LatticeECP).
Select Pipelining	**Enable Input Registers** – Enables the input registers. This option is selected if any input is selected as a “Shift” input. **Enable Pipeline Register** - Enables the pipeline registers at the output of each multiplier. **Enable Output Register** - Enables the output register after the ADDER. **RST on all registers** - Connects a reset pin to all registers of the DSP block. **CE on all registers** - Connects a clock enable pin to all registers of the DSP block.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 218: MULTADDSUB Dialog Box for LatticeECP, LatticeECP2/M, and LatticeXP2—Advanced Tab

Feature	Description
Select Pipelining	**Enable Input Register A0, A1, B0, B1** – Specify options for Input registers. This option is disabled if the Enable Input Registers option on the Basic tab is not selected. **Enable Signed Register A, B –** (not available in LatticeECP) Specify options for signed registers A or B of the DSP block. This option is disabled unless the Signed/ Unsigned A, B option on the Basic tab is set to Dynamic and the Enable Input Registers option is selected. **Enable Signed Pipeline Register A, B –** (not available in LatticeECP) Specify options for signed pipeline registers A or B of the DSP block. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Addsub Input Register** – Specify options for the ADDNSUB input register. This option is enabled only if the ADDER is set to dynamic mode. **Enable Addsub Pipeline Register** – Specify options for the ADDNSUB pipeline register. This option is enabled only if the ADDER is set to dynamic mode. **Enable Multiplier Pipeline Register 0, 1** – (not available in LatticeECP) Specify options for the pipeline registers of multiplier 0 or 1. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Pipeline Register 0, 1** – (LatticeECP only) Specify options for the pipeline registers of multiplier 0 or 1. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Output Register** – Specify options for the output registers. If the ADDER is implemented in the PFU, this option is used for the multiplier blocks. There will be no register after the ADDER block. The ADDER will be built using PFU logic if the module does not fit into a single DSP element. This option is disabled if the Enable Output Register option on the Basic tab is not selected.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*

pmi_dsp_multaddsub

The following table is intended to provide information for pmi_dsp_multaddsub usage (pmi_dsp_multaddsub.v).

pmi_dsp_multaddsub Control and Data Signal Descriptions

RST Asynchronous reset of selected registers		
SignA Dynamic signal:	0 = unsigned,	1 = signed
SignB Dynamic signal:	0 = unsigned	1 = signed
ACCUMSLOAD Dynamic signal:	0 = accumulate	1 = load
ADDNSUB Dynamic signal:	0 = subtract	1 = add
SOURCEA Dynamic signal:	0 = parallel input	1 = shift input
SOURCEB Dynamic signal:	0 = parallel input	1 = shift input

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_dsp_multaddsub.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	CLK0	Bit	N/A
I	CLK1	Bit	N/A
I	CLK2	Bit	N/A
I	CLK3	Bit	N/A
I	CE0	Bit	N/A
I	CE1	Bit	N/A
I	CE2	Bit	N/A
I	CE3	Bit	N/A
I	RST0	Bit	N/A
I	RST1	Bit	N/A
I	RST2	Bit	N/A
I	RST3	Bit	N/A
I	SignA	Bit	N/A
I	SignB	Bit	N/A
I	ShiftA0	Bit	N/A
I	ShiftA1	Bit	N/A
I	ShiftB0	Bit	N/A
I	ShiftB1	Bit	N/A
I	ADDNSUB	Bit	N/A
I	A0	Bus	(pmi_dataa_width -1):0
I	SRIA	Bus	17:0
I	A1	Bus	(pmi_dataa_width -1):0
	B0	Bus	(pmi_datab_width - 1):0
I	SRIB	Bus	17:0
I	B1	Bus	(pmi_datab_width - 1):0
O	SUM	Bus	(pmi_dataa_width + pmi_datab_width):0
O	SROA	Bus	17:0
O	SROB	Bus	17:0

Verilog pmi_dsp_multaddsub Definition

```
module pmi_dsp_multaddsub
 #(parameter pmi_dataa_width = 8,
```

```
 parameter pmi_datab_width = 8,
 parameter pmi_additional_pipeline = 0,
 parameter pmi_input_reg = "on",
 parameter pmi_output_reg = "on",
 parameter pmi_family = "ECP2",
 parameter pmi_gsr = "disable",
 parameter pmi_source_control_a0 = "parallel",
 parameter pmi_source_control_a1 = "parallel",
 parameter pmi_source_control_b0 = "parallel",
 parameter pmi_source_control_b1 = "parallel",
 parameter pmi_reg_inputa0_clk = "CLK0",
 parameter pmi_reg_inputa0_ce = "CE0",
 parameter pmi_reg_inputa0_rst = "RST0",
 parameter pmi_reg_inputa1_clk = "CLK0",
 parameter pmi_reg_inputa1_ce = "CE0",
 parameter pmi_reg_inputa1_rst = "RST0",
 parameter pmi_reg_inputb0_clk = "CLK0",
 parameter pmi_reg_inputb0_ce = "CE0",
 parameter pmi_reg_inputb0_rst = "RST0",
 parameter pmi_reg_inputb1_clk = "CLK0",
 parameter pmi_reg_inputb1_ce = "CE0",
 parameter pmi_reg_inputb1_rst = "RST0",
 parameter pmi_reg_pipeline0_clk = "CLK0",
 parameter pmi_reg_pipeline0_ce = "CE0",
 parameter pmi_reg_pipeline0_rst = "RST0",
 parameter pmi_reg_pipeline1_clk = "CLK0",
 parameter pmi_reg_pipeline1_ce = "CE0",
 parameter pmi_reg_pipeline1_rst = "RST0",
 parameter pmi_reg_output_clk = "CLK0",
 parameter pmi_reg_output_ce = "CE0",
 parameter pmi_reg_output_rst = "RST0",
 parameter pmi_reg_signeda_0_clk = "CLK0",
 parameter pmi_reg_signeda_0_ce = "CE0",
 parameter pmi_reg_signeda_0_rst = "RST0",
 parameter pmi_reg_signeda_1_clk = "CLK0",
 parameter pmi_reg_signeda_1_ce = "CE0",
 parameter pmi_reg_signeda_1_rst = "RST0",
 parameter pmi_reg_signedb_0_clk = "CLK0",
 parameter pmi_reg_signedb_0_ce = "CE0",
 parameter pmi_reg_signedb_0_rst = "RST0",
 parameter pmi_reg_signedb_1_clk = "CLK0",
 parameter pmi_reg_signedb_1_ce = "CE0",
 parameter pmi_reg_signedb_1_rst = "RST0",
 parameter pmi_reg_addnsub_0_clk = "CLK0",
 parameter pmi_reg_addnsub_0_ce = "CE0",
 parameter pmi_reg_addnsub_0_rst = "RST0",
 parameter pmi_reg_addnsub_1_clk = "CLK0",
 parameter pmi_reg_addnsub_1_ce = "CE0",
 parameter pmi_reg_addnsub_1_rst = "RST0",
 parameter module_type = "pmi_dsp_multaddsub")

(input [(pmi_dataa_width-1):0]       A0, A1,
 input [(pmi_datab_width-1):0]       B0, B1,
 input [17:0]       SRIA,
 input [17:0]       SRIB,
 input CLK0,
 input CLK1,
 input CLK2,
 input CLK3,
```

```
    input CE0,
    input CE1,
    input CE2,
    input CE3,
    input RST0,
    input RST1,
    input RST2,
    input RST3,
    input SignA,
    input SignB,
    input ShiftA0,
    input ShiftA1,
    input ShiftB0,
    input ShiftB1,
    input ADDNSUB,
    output [(pmi_dataa_width + pmi_datab_width):0]  SUM,
    output [17:0]  SROA,
    output [17:0]  SROB)/*synthesis syn_black_box */;

endmodule // pmi_dsp_multaddsub
```

VHDL pmi_dsp_multaddsub Definition

```
component pmi_dsp_multaddsub is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       pmi_additional_pipeline : integer := 0;
       pmi_input_reg : string := "on";
       pmi_output_reg: string := "on";
       pmi_family : string := "ECP2";
       pmi_gsr : string := "disable";
       pmi_source_control_a0 : string := "parallel";
       pmi_source_control_a1 : string := "parallel";
       pmi_source_control_b0 : string := "parallel";
       pmi_source_control_b1 : string := "parallel";
       pmi_reg_inputa0_clk : string := "CLK0";
       pmi_reg_inputa0_ce : string := "CE0";
       pmi_reg_inputa0_rst : string := "RST0";
       pmi_reg_inputa1_clk : string := "CLK0";
       pmi_reg_inputa1_ce : string := "CE0";
       pmi_reg_inputa1_rst : string := "RST0";
       pmi_reg_inputb0_clk : string := "CLK0";
       pmi_reg_inputb0_ce : string := "CE0";
       pmi_reg_inputb0_rst : string := "RST0";
       pmi_reg_inputb1_clk : string := "CLK0";
       pmi_reg_inputb1_ce : string := "CE0";
       pmi_reg_inputb1_rst : string := "RST0";
       pmi_reg_pipeline0_clk : string := "CLK0";
       pmi_reg_pipeline0_ce : string := "CE0";
       pmi_reg_pipeline0_rst : string := "RST0";
       pmi_reg_pipeline1_clk : string := "CLK0";
       pmi_reg_pipeline1_ce : string := "CE0";
       pmi_reg_pipeline1_rst : string := "RST0";
       pmi_reg_output_clk : string := "CLK0";
       pmi_reg_output_ce : string := "CE0";
       pmi_reg_output_rst : string := "RST0";
       pmi_reg_signeda_0_clk : string := "CLK0";
       pmi_reg_signeda_0_ce : string := "CE0";
```

```
        pmi_reg_signeda_0_rst : string := "RST0";
        pmi_reg_signeda_1_clk : string := "CLK0";
        pmi_reg_signeda_1_ce : string := "CE0";
        pmi_reg_signeda_1_rst : string := "RST0";
        pmi_reg_signedb_0_clk : string := "CLK0";
        pmi_reg_signedb_0_ce : string := "CE0";
        pmi_reg_signedb_0_rst : string := "RST0";
        pmi_reg_signedb_1_clk : string := "CLK0";
        pmi_reg_signedb_1_ce: string := "CE0";
        pmi_reg_signedb_1_rst : string := "RST0";
        pmi_reg_addnsub_0_clk : string := "CLK0";
        pmi_reg_addnsub_0_ce : string := "CE0";
        pmi_reg_addnsub_0_rst : string := "RST0";
        pmi_reg_addnsub_1_clk : string := "CLK0";
        pmi_reg_addnsub_1_ce : string := "CE0";
        pmi_reg_addnsub_1_rst : string := "RST0";
        module_type : string := "pmi_dsp_multaddsub"
    );
    port (
     A0, A1:                 in
std_logic_vector((pmi_dataa_width-1) downto 0);
     B0, B1:                 in
std_logic_vector((pmi_datab_width-1) downto 0);
     SRIA,SRIB:              in std_logic_vector(17 downto 0);
     CLK0, CLK1, CLK2, CLK3: in std_logic;
     CE0, CE1, CE2, CE3:     in std_logic;
     RST0, RST1, RST2, RST3: in std_logic;
     SignA, SignB:           in std_logic;
     ShiftA0, ShiftA1:       in std_logic;
     ShiftB0, ShiftB1:       in std_logic;
     ADDNSUB:                in std_logic;
     SUM:                    out
std_logic_vector((pmi_dataa_width + pmi_datab_width) downto 0);
     SROA:                   out std_logic_vector(17 downto 0);
     SROB:                   out std_logic_vector(17 downto 0)
   );
  end component pmi_dsp_multaddsub;
```

Parameter Names and Values

The following table contains the pmi_dsp_multaddsub module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 72	8	
pmi_datab_width	2 to 72	8	
pmi_gsr	“enable” | “disable”	“enable”	
pmi_additional_pipeline	0 to 1	0	
pmi_input_reg	“on” | “off”	“on”	
pmi_output_reg	“on” | “off”	“on”	
pmi_family	“ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “XP2”	“ECP2”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_source_controla0	“parallel”	“shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controlb0 and pmi_source_controlb1 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controla1	“parallel”	“shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controlb0 and pmi_source_controlb1 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controlb0	“parallel”	“shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controlb0 and pmi_source_controlb1 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controlb1	“parallel”	“shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controlb0 and pmi_source_controlb1 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_reg_inputa0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_inputa0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_inputa0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_inputa1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_inputa1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_inputa1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_inputb0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_inputb0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_inputb0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_inputb1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_inputb1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_inputb1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_reg_pipeline0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_pipeline0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_pipeline0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_pipeline1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_pipeline1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_pipeline1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_output_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_output_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_output_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_signeda_0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_signeda_0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_signeda_0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_signeda_1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_signeda_1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_signeda_1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_signedb_0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_signedb_0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_signedb_0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_signedb_1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_signedb_1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_signedb_1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_addnsub_0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_addnsub_0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_reg_addnsub_0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_addnsub_1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_addnsub_1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_addnsub_1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

pmi_dsp_casmultaddsub

The following table is intended to provide information for pmi_dsp_casmultaddsub usage (pmi_dsp_multaddsub.v). This module is for LatticeECP3.

pmi_dsp_casmultaddsub Control and Data Signal Descriptions

RST Asynchronous reset of selected registers		
SignA Dynamic signal:	0 = unsigned	1 = signed
SignB Dynamic signal:	0 = unsigned	1 = signed
ADDNSUB Dynamic signal:	0 = subtract	1 = add

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_dsp_casmultaddsub.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	CLK0	Bit	N/A
I	CLK1	Bit	N/A
I	CLK2	Bit	N/A
I	CLK3	Bit	N/A
I	CE0	Bit	N/A
I	CE1	Bit	N/A
I	CE2	Bit	N/A
I	CE3	Bit	N/A
I	RST0	Bit	N/A
I	RST1	Bit	N/A
I	RST2	Bit	N/A
I	RST3	Bit	N/A
I	SignA	Bit	N/A
I	SignB	Bit	N/A
I	ShiftA0	Bit	N/A
I	ShiftA1	Bit	N/A
I	ShiftB0	Bit	N/A
I	ShiftB1	Bit	N/A
I	ADDNSUB	Bit	N/A
I	SignCIN	Bit	N/A
I	A0	Bus	(pmi_dataa_width -1):0
I	SRIA	Bus	17:0
I	A1	Bus	(pmi_dataa_width -1):0
I	B0	Bus	(pmi_datab_width - 1):0
I	SRIB	Bus	17:0
I	B1	Bus	(pmi_datab_width - 1):0
I	CIN	Bus	53:0
O	SUM	Bus	53:0
O	SROA	Bus	17:0
O	SROB	Bus	17:0

Verilog pmi_dsp_casmultaddsub Definition

```
module pmi_dsp_casmultaddsub
 #(parameter pmi_dataa_width = 18,
   parameter pmi_datab_width = 18,
   parameter pmi_additional_pipeline = 0,
   parameter pmi_input_reg = "on",
   parameter pmi_output_reg = "on",
   parameter pmi_family = "ECP3",
   parameter pmi_gsr = "enable",
   parameter pmi_reg_inputa0_clk = "CLK0",
   parameter pmi_reg_inputa0_ce = "CE0",
   parameter pmi_reg_inputa0_rst = "RST0",
   parameter pmi_reg_inputa1_clk = "CLK0",
   parameter pmi_reg_inputa1_ce = "CE0",
   parameter pmi_reg_inputa1_rst = "RST0",
   parameter pmi_reg_inputb0_clk = "CLK0",
   parameter pmi_reg_inputb0_ce = "CE0",
   parameter pmi_reg_inputb0_rst = "RST0",
   parameter pmi_reg_inputb1_clk = "CLK0",
   parameter pmi_reg_inputb1_ce = "CE0",
   parameter pmi_reg_inputb1_rst = "RST0",
   parameter pmi_reg_pipeline0_clk = "CLK0",
   parameter pmi_reg_pipeline0_ce = "CE0",
   parameter pmi_reg_pipeline0_rst = "RST0",
   parameter pmi_reg_pipeline1_clk = "CLK0",
   parameter pmi_reg_pipeline1_ce = "CE0",
   parameter pmi_reg_pipeline1_rst = "RST0",
   parameter pmi_reg_output_clk = "CLK0",
   parameter pmi_reg_output_ce = "CE0",
   parameter pmi_reg_output_rst = "RST0",
   parameter pmi_reg_signeda_0_clk = "CLK0",
   parameter pmi_reg_signeda_0_ce = "CE0",
   parameter pmi_reg_signeda_0_rst = "RST0",
   parameter pmi_reg_signeda_1_clk = "CLK0",
   parameter pmi_reg_signeda_1_ce = "CE0",
   parameter pmi_reg_signeda_1_rst = "RST0",
   parameter pmi_reg_signedb_0_clk = "CLK0",
   parameter pmi_reg_signedb_0_ce = "CE0",
   parameter pmi_reg_signedb_0_rst = "RST0",
   parameter pmi_reg_signedb_1_clk = "CLK0",
   parameter pmi_reg_signedb_1_ce = "CE0",
   parameter pmi_reg_signedb_1_rst = "RST0",
   parameter pmi_reg_addnsub_0_clk = "CLK0",
   parameter pmi_reg_addnsub_0_ce = "CE0",
   parameter pmi_reg_addnsub_0_rst = "RST0",
   parameter pmi_reg_addnsub_1_clk = "CLK0",
   parameter pmi_reg_addnsub_1_ce = "CE0",
   parameter pmi_reg_addnsub_1_rst = "RST0",
   parameter module_type = "pmi_dsp_casmultaddsub")

  (input [(pmi_dataa_width-1):0]       A0, A1,
   input [(pmi_datab_width-1):0]       B0, B1,
   input [17:0]       SRIA,
   input [17:0]       SRIB,
   input CLK0,
   input CLK1,
   input CLK2,
   input CLK3,
   input CE0,
   input CE1,
```

```
    input CE2,
    input CE3,
    input RST0,
    input RST1,
    input RST2,
    input RST3,
    input SignA,
    input SignB,
    input ShiftA0,
    input ShiftA1,
    input ShiftB0,
    input ShiftB1,
    input ADDNSUB,
    input SignCIN,
    input [53:0] CIN,
    output SignSUM,
    output [53:0]  SUM,
    output [17:0]  SROA,
    output [17:0]  SROB)/*synthesis syn_black_box */;

endmodule // pmi_dsp_casmultaddsub
```

VHDL pmi_dsp_casmultaddsub Definition

```
component pmi_dsp_casmultaddsub is
     generic (
       pmi_dataa_width : integer := 18;
       pmi_datab_width : integer := 18;
       pmi_additional_pipeline : integer := 0;
       pmi_input_reg : string := "on";
       pmi_output_reg : string := "on";
       pmi_family : string := "ECP3";
       pmi_gsr : string := "enable";
       pmi_reg_inputa0_clk : string := "CLK0";
       pmi_reg_inputa0_ce : string := "CE0";
       pmi_reg_inputa0_rst : string := "RST0";
       pmi_reg_inputa1_clk : string := "CLK0";
       pmi_reg_inputa1_ce : string := "CE0";
       pmi_reg_inputa1_rst : string := "RST0";
       pmi_reg_inputb0_clk : string := "CLK0";
       pmi_reg_inputb0_ce : string := "CE0";
       pmi_reg_inputb0_rst : string := "RST0";
       pmi_reg_inputb1_clk : string := "CLK0";
       pmi_reg_inputb1_ce : string := "CE0";
       pmi_reg_inputb1_rst : string := "RST0";
       pmi_reg_pipeline0_clk : string := "CLK0";
       pmi_reg_pipeline0_ce : string := "CE0";
       pmi_reg_pipeline0_rst : string := "RST0";
       pmi_reg_pipeline1_clk : string := "CLK0";
       pmi_reg_pipeline1_ce : string := "CE0";
       pmi_reg_pipeline1_rst : string := "RST0";
       pmi_reg_output_clk : string := "CLK0";
       pmi_reg_output_ce : string := "CE0";
       pmi_reg_output_rst : string := "RST0";
       pmi_reg_signeda_0_clk : string := "CLK0";
       pmi_reg_signeda_0_ce : string := "CE0";
       pmi_reg_signeda_0_rst : string := "RST0";
       pmi_reg_signeda_1_clk : string := "CLK0";
       pmi_reg_signeda_1_ce : string := "CE0";
```

```
        pmi_reg_signeda_1_rst : string := "RST0";
        pmi_reg_signedb_0_clk : string := "CLK0";
        pmi_reg_signedb_0_ce : string := "CE0";
        pmi_reg_signedb_0_rst : string := "RST0";
        pmi_reg_signedb_1_clk : string := "CLK0";
        pmi_reg_signedb_1_ce : string := "CE0";
        pmi_reg_signedb_1_rst : string := "RST0";
        pmi_reg_addnsub_0_clk : string := "CLK0";
        pmi_reg_addnsub_0_ce : string := "CE0";
        pmi_reg_addnsub_0_rst : string := "RST0";
        pmi_reg_addnsub_1_clk : string := "CLK0";
        pmi_reg_addnsub_1_ce : string := "CE0";
        pmi_reg_addnsub_1_rst : string := "RST0";
        module_type : string := "pmi_dsp_casmultaddsub"
      );
      port (
        A0, A1 : in std_logic_vector((pmi_dataa_width-1) downto
0);
        B0, B1 : in std_logic_vector((pmi_datab_width-1) downto
0);
        SRIA : in std_logic_vector(17 downto 0);
        SRIB : in std_logic_vector(17 downto 0);
        CLK0 : in std_logic;
        CLK1 : in std_logic;
        CLK2 : in std_logic;
        CLK3 : in std_logic;
        CE0 : in std_logic;
        CE1 : in std_logic;
        CE2 : in std_logic;
        CE3 : in std_logic;
        RST0 : in std_logic;
        RST1 : in std_logic;
        RST2 : in std_logic;
        RST3 : in std_logic;
        SignA : in std_logic;
        SignB : in std_logic;
        ShiftA0 : in std_logic;
        ShiftA1 : in std_logic;
        ShiftB0 : in std_logic;
        ShiftB1 : in std_logic;
        ADDNSUB : in std_logic;
        SignCIN : in std_logic;
        CIN : in std_logic_vector(53 downto 0);
        SignSUM : out std_logic;
        SUM : out std_logic_vector(53 downto 0);
        SROA : out std_logic_vector(17 downto 0);
        SROB : out std_logic_vector(17 downto 0)
      );
    end component pmi_dsp_casmultaddsub;
```

Parameter Names and Values

The following table contains the pmi_dsp_casmultaddsub module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 18	18	
pmi_datab_width	2 to 18	18	
pmi_gsr	“enable” | “disable”	“enable”	
pmi_additional_pipeline	0 to 1	0	
pmi_input_reg	“on” | “off”	“on”	
pmi_output_reg	“on” | “off”	“on”	
pmi_family	“ECP3” | “ECP5U” | “ECP5UM”	“ECP3”	
pmi_reg_inputa0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputa0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputa0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputa1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputa1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputa1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputb0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputb0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputb0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputb1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputb1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputb1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_pipeline0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_pipeline0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_pipeline0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_pipeline1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_reg_pipeline1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_pipeline1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_output_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_output_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_output_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signeda_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signeda_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signeda_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signeda_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signeda_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signeda_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signedb_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signedb_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signedb_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signedb_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signedb_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signedb_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_addnsub_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_addnsub_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_addnsub_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	

See Also

- "Use PMI or IPexpress?" on page 191
- "Using PMI" on page 200

MULTADDSUBSUM

Two-input MULTADDSUBSUM block multiplier that provides a variety of multiplier, add/subtract, and sum organizations and is implemented as embedded sysDSP Blocks. Ports, properties, and functionality are described in detail in the technical notes listed in the "See Also" section below.

The options vary with the device family. See:

- "MULTADDSUBSUM Options (ECP5, LatticeECP3)" on page 1526
- "MULTADDSUBSUM Options (LatticeECP)" on page 1527
- "MULTADDSUBSUM Options (LatticeECP2/M, LatticeXP2)" on page 1529

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysDSP" on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*

MULTADDSUBSUM Options (ECP5, LatticeECP3)

Table 219: MULTADDSUBSUM Dialog Box for ECP5 and LatticeECP3

Feature	Description
Size of DSPMULTADDSUBSUM block	**Input A0/A1/A2/A3 width** – Specify the size of the first term in the multiplication (for all multipliers). **Input B0/B1/B2/B3 width** – Specify the size of the second term in the multiplication (for all multipliers). **Sum output width** – Size of the output. This is two more than the product size. That is, it is equal to A size + B size + 2.
Select Options	**Select Shift Out A** - Enables the shift-out port A of the DSP block. **Select Shift Out B** - Enables the shift-out port B of the DSP block. **Add/Sub 0, 1 Operation** - Select if the adder is in add, subtract, or dynamic mode. In dynamic mode high = add, low = subtract.

Table 219: MULTADDSUBSUM Dialog Box for ECP5 and LatticeECP3 (Continued)

Feature	Description
Data Options	**Input A0, A1, A2, A3, B0, B1, B2, B3 Source** – Specify if the source of the input is Parallel for generic routing, Shift for the shift outputs of an adjacent DSP block, or Dynamic for dynamic selection of the source of the input. **Data Type A, B** – Specify the data format of the inputs as signed, unsigned, or dynamic selection of the sign.
Register Options	**Reset Mode** – Specify whether the reset is synchronous or asynchronous. **Clock** – Selects the clock for the registers of the DSP block. **CE** – Selects the clock enable pin to the registers of the DSP block. **RST** – Selects the reset pin to the registers of the DSP block.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysDSP" on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*

MULTADDSUBSUM Options (LatticeECP)

The module dialog box has two tabs:

- Basic, below
- Advanced, see Table 221 on page 1529

In the Basic tab, you can enable pipelining and set parameters such as the size and format of the data. In the Advanced tab, you can control each group of pipelining registers with independent clocks, clock enables, and resets.

Table 220: MULTADDSUBSUM Dialog Box for LatticeECP—Basic Tab

Feature	Description
Size of the DSPMADDSUM block	**Multiplicand A0/A1/A2/A3 bit size** – Specify the size of the first term in the multiplication (for all multipliers). **Multiplier B0/B1/B2/B3 bit size** – Specify the size of the second term in the multiplication (for all multipliers). **Sum output bit size** – Size of the output. This is 2 more than the product size. That is, it is equal to A size + B size + 2.
Select Block Options	**Area/Speed** - Specify if the implementation should minimize the number of DSP blocks (Area) or minimize number of partial products to be added (Speed). In some cases, both options will produce the same implementation. **Add/Sub 1, 3 Operation** - Select if the ADDER is in add, subtract, or dynamic mode. In dynamic mode high = add, low = subtract.
Data Format	**Note**: The shift options, input and out, are not available if either of the A and B inputs is larger than 18 bits. **Input A0, A1, A2, A3** – Specify if the source of input A for multiplier 0, 1, 2, or 3 is from generic routing (Parallel) or the shift outputs of the adjacent DSP block (Shift). **Select Shift Out A** - Enables the shift-out port A of the DSP block. **Input B0, B1, B2, B3** – Specify if the source of input B for multiplier 0, 1, 2, or 3 is from generic routing (Parallel) or the shift outputs of the adjacent DSP block (Shift). **Select Shift Out B** - Enables the shift-out port B of the DSP block. **Signed/Unsigned** - Specify if the data format of the inputs is signed or unsigned.
Select Pipelining	**Enable Input Registers** – Enables the input registers. **Enable Pipeline Register** - Enables the pipeline registers. **Enable Output Register** - Enables the output registers. **RST on all registers** - Connects a reset pin to all registers of the DSP block. **CE on all registers** - Connects a clock enable pin to all registers of the DSP block.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 221: MULTADDSUBSUM Dialog Box for LatticeECP—Advanced Tab

Feature	Description
Select Pipelining	**Enable Input Register A0, A1, A2, A3, B0, B1, B2, B3** – Specify options for Input registers. This option is disabled if the Enable Input Registers option on the Basic tab is not selected. **Enable Addsub1 Input Register** – Specify options for the ADDNSUB1 input register. This option is available only if the ADDER is set to dynamic mode. **Enable Addsub1 Pipeline Register** – Specify options for the ADDNSUB1 pipeline register. This option is enabled only if the ADDER is set to dynamic mode. **Enable Addsub3 Input Register** – Specify options for the ADDNSUB3 input register. This option is available only if the ADDER is set to dynamic mode. **Enable Addsub3 Pipeline Register** – Specify options for the ADDNSUB3 pipeline register. This option is available only if the ADDER is set to dynamic mode. **Enable Pipeline Register 0, 1, 2, 3** – Specify options for the pipeline registers of multipliers 0, 1, 2, or 3. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Output Register** – Specify options for the output registers. If the Adders and Summer are implemented in the PFU, this option is used for the multiplier blocks. There will be no register after the ADDSUB/SUM block. If the size of the module exceeds one DSP element, the ADDERs and SUM will be implemented in PFU. This option is disabled if the Enable Output Register option on the Basic tab is not selected.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified and saves the files in the project folder.
Close	Cancels the dialog box without generating the module.
Help	Opens the Help Contents.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172
- TN1057, *LatticeECP-DSP sysDSP Usage Guide*

MULTADDSUBSUM Options (LatticeECP2/M, LatticeXP2)

The module dialog box has three tabs:

- Basic, below
- Advanced, see Table 223 on page 1531
- Advanced Cont., see Table 224 on page 1532

In the Basic tab you set most of the parameters including the size and format of the data, and enable pipelining. In the Advanced tab, you can control each group of pipelining registers with independent clocks, clock enables, and resets. Because of the number of registers, the advanced controls are on two tabs, Advanced and Advanced Cont.

Table 222: MULTADDSUBSUM Dialog Box for LatticeECP2/M, and LatticeXP2—Basic Tab

Feature	Description
Size of the DSPMADDSUM block	**Multiplicand A0/A1/A2/A3 bit size** – Specify the size of the first term in the multiplication (for all multipliers). **Multiplier B0/B1/B2/B3 bit size** – Specify the size of the second term in the multiplication (for all multipliers). **Sum output bit size** – Size of the output. This is 2 more than the product size. That is, it is equal to A size + B size + 2.
Select Block Options	**Area/Speed** - Specify if the implementation should minimize the number of DSP blocks (Area) or minimize number of partial products to be added (Speed). In some cases, both options will produce the same implementation. **Add/Sub Implementation -** Specify if the ADDER is built using a DSP block or Look Up Table (LUT) logic. **Sum Implementation -** Specify if this function is built using a DSP block or Look Up Table (LUT) logic. **Add/Sub 1, 3 Operation** - Select if the ADDER is in add, subtract, or dynamic mode. In dynamic mode high = add, low = subtract.
Data Format	**Note**: The shift options, input and out, are not available if either of the A and B inputs is larger than 18 bits. **Input A0, A1, A2, A3** – Specify if the source of input A for multiplier 0, 1, 2, or 3 is from generic routing (Parallel) or the shift outputs of the adjacent DSP block (Shift). The source can also have dynamic control. **Select Shift Out A** - Enables the shift-out port A of the DSP block. **Input B0, B1, B2, B3** – Specify if the source of input B for multiplier 0, 1, 2, or 3 is from generic routing (Parallel) or the shift outputs of the adjacent DSP block (Shift). The source can also have dynamic control. **Select Shift Out B** - Enables the shift-out port B of the DSP block. **Signed/Unsigned A, B** - Specify if the data format of the inputs is signed or unsigned. The data can also have dynamic control. **Special sign extension for unsigned subtract** - This option is available only when one or both of the add/sub operations is in subtract or dynamic mode and both of the Signed/Unsigned A and B options are set to Unsigned.
Select Pipelining	**Enable Input Registers** – Enables the input registers. **Enable Pipeline Registers** - Enables the pipeline registers. **Enable Output Register** - Enables the output registers. **RST on all registers** - Connects a reset pin to all registers of the DSP block. **CE on all registers** - Connects a clock enable pin to all registers of the DSP block.

Table 222: MULTADDSUBSUM Dialog Box for LatticeECP2/M, and LatticeXP2—Basic Tab

Feature	Description
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 223: MULTADDSUBSUM Dialog Box for LatticeECP2/M, and LatticeXP2—Advanced Tab

Feature	Description
Select Pipelining	**Enable Input Register A0, A1, A2, A3, B0, B1, B2, B3** – Specify options for Input registers. This option is disabled if the Enable Input Registers option on the Basic tab is not selected. **Enable Signed Register A, B –** Specify options for signed registers A or B of the DSP block. This option is disabled unless the Signed/Unsigned A, B option on the Basic tab is set to Dynamic and the Enable Input Registers option is selected. **Enable Signed Pipeline Register A, B –** Specify options for signed pipeline registers A or B of the DSP block. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 224: MULTADDSUBSUM Dialog Box for LatticeECP2/M, and LatticeXP2—Advanced Cont. Tab

Feature	Description
Select Pipelining	**Enable Addsub1 Input Register** – Specify options for the ADDNSUB1 input register. This option is enabled only if the ADDER is set to dynamic mode. **Enable Addsub1 Pipeline Register** – Specify options for the ADDNSUB1 pipeline register. This option is enabled only if the ADDER is set to dynamic mode. **Enable Addsub3 Input Register** – Specify options for the ADDNSUB3 input register. This option is enabled only if the ADDER is set to dynamic mode. **Enable Addsub3 Pipeline Register** – Specify options for the ADDNSUB3 pipeline register. This option is enabled only if the ADDER is set to dynamic mode. **Enable Multiplier Pipeline Register 0, 1, 2, 3** – Specify options for the pipeline registers of multipliers 0, 1, 2, or 3. This option is disabled if the Enable Pipeline Registers option on the Basic tab is not selected. **Enable Output Register** – Specify options for the output registers. If the Adders and Summer are implemented in the PFU, this option is used for the multiplier blocks. There will be no register after the ADDSUB/SUM block. If the size of the module exceeds one DSP element, the ADDERs and SUM will be implemented in PFU. This option is disabled if the Enable Output Register option on the Basic tab is not selected.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysDSP” on page 1172
- TN1107, *LatticeECP2/M sysDSP Usage Guide*
- TN1140, *LatticeXP2 sysDSP Usage Guide*

pmi_dsp_multaddsubsum

The following table is intended to provide information for pmi_dsp_multaddsubsum usage (pmi_dsp_multaddsubsum.v).

pmi_dsp_multaddsubsum Control and Data Signal Descriptions

RST Asynchronous reset of selected registers		
SignA Dynamic signal:	0 = unsigned	1 = signed
SignB Dynamic signal:	0 = unsigned	1 = signed
ADDNSUB1 Dynamic signal:	0 = subtract	1 = add
ADDNSUB3 Dynamic signal:	0 = subtract	1 = add

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_dsp_multaddsubsum.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	CLK0	Bit	N/A
I	CLK1	Bit	N/A
I	CLK2	Bit	N/A
I	CLK3	Bit	N/A
I	CE0	Bit	N/A
I	CE1	Bit	N/A
I	CE2	Bit	N/A
I	CE3	Bit	N/A
I	RST0	Bit	N/A
I	RST1	Bit	N/A
I	RST2	Bit	N/A
I	RST3	Bit	N/A
I	SignA	Bit	N/A
I	SignB	Bit	N/A
I	ShiftA0	Bit	N/A
I	ShiftA1	Bit	N/A
I	ShiftA2	Bit	N/A
I	ShiftA3	Bit	N/A
I	ShiftB0	Bit	N/A
I	ShiftB1	Bit	N/A
I	ShiftB2	Bit	N/A
I	ShiftB3	Bit	N/A
I	ADDNSUB1	Bit	N/A
I	ADDNSUB3	Bit	N/A
i	A0	Bus	(pmi_dataa_width - 1):0
I	SRIA	Bus	17:0
I	A1	Bus	(pmi_dataa_width - 1):0
I	A2	Bus	(pmi_dataa_width - 1):0
I	A3	Bus	(pmi_dataa_width - 1):0
I	B0	Bus	(pmi_datab_width - 1):0
I	SRIB	Bus	17:0

Input/Output	Port Name	Type	Size (Buses Only)
I	B1	Bus	(pmi_datab_width - 1):0
I	B2	Bus	(pmi_datab_width - 1):0
I	B3	Bus	(pmi_datab_width - 1):0
O	SUM	Bus	(pmi_dataa_width + pmi_datab_width + 1):0
O	SROA	Bus	17:0
O	SROB	Bus	17:0

Verilog pmi_dsp_multaddsubsum Definition

```
module pmi_dsp_multaddsubsum
 #(parameter pmi_dataa_width = 8,
   parameter pmi_datab_width = 8,
   parameter pmi_additional_pipeline = 0,
   parameter pmi_input_reg = "on",
   parameter pmi_output_reg = "on",
   parameter pmi_family = "ECP2",
   parameter pmi_gsr = "disable",
   parameter pmi_source_control_a0 = "parallel",
   parameter pmi_source_control_a1 = "parallel",
   parameter pmi_source_control_a2 = "parallel",
   parameter pmi_source_control_a3 = "parallel",
   parameter pmi_source_control_b0 = "parallel",
   parameter pmi_source_control_b1 = "parallel",
   parameter pmi_source_control_b2 = "parallel",
   parameter pmi_source_control_b3 = "parallel",
   parameter pmi_reg_inputa0_clk = "CLK0",
   parameter pmi_reg_inputa0_ce = "CE0",
   parameter pmi_reg_inputa0_rst = "RST0",
   parameter pmi_reg_inputa1_clk = "CLK0",
   parameter pmi_reg_inputa1_ce = "CE0",
   parameter pmi_reg_inputa1_rst = "RST0",
   parameter pmi_reg_inputa2_clk = "CLK0",
   parameter pmi_reg_inputa2_ce = "CE0",
   parameter pmi_reg_inputa2_rst = "RST0",
   parameter pmi_reg_inputa3_clk = "CLK0",
   parameter pmi_reg_inputa3_ce = "CE0",
   parameter pmi_reg_inputa3_rst = "RST0",
   parameter pmi_reg_inputb0_clk = "CLK0",
   parameter pmi_reg_inputb0_ce = "CE0",
   parameter pmi_reg_inputb0_rst = "RST0",
   parameter pmi_reg_inputb1_clk = "CLK0",
   parameter pmi_reg_inputb1_ce = "CE0",
   parameter pmi_reg_inputb1_rst = "RST0",
   parameter pmi_reg_inputb2_clk = "CLK0",
   parameter pmi_reg_inputb2_ce = "CE0",
   parameter pmi_reg_inputb2_rst = "RST0",
   parameter pmi_reg_inputb3_clk = "CLK0",
   parameter pmi_reg_inputb3_ce = "CE0",
   parameter pmi_reg_inputb3_rst = "RST0",
   parameter pmi_reg_pipeline0_clk = "CLK0",
   parameter pmi_reg_pipeline0_ce = "CE0",
   parameter pmi_reg_pipeline0_rst = "RST0",
```

```
 parameter pmi_reg_pipeline1_clk = "CLK0",
 parameter pmi_reg_pipeline1_ce = "CE0",
 parameter pmi_reg_pipeline1_rst = "RST0",
 parameter pmi_reg_pipeline2_clk = "CLK0",
 parameter pmi_reg_pipeline2_ce = "CE0",
 parameter pmi_reg_pipeline2_rst = "RST0",
 parameter pmi_reg_pipeline3_clk = "CLK0",
 parameter pmi_reg_pipeline3_ce = "CE0",
 parameter pmi_reg_pipeline3_rst = "RST0",
 parameter pmi_reg_output_clk = "CLK0",
 parameter pmi_reg_output_ce = "CE0",
 parameter pmi_reg_output_rst = "RST0",
 parameter pmi_reg_signeda_0_clk = "CLK0",
 parameter pmi_reg_signeda_0_ce = "CE0",
 parameter pmi_reg_signeda_0_rst = "RST0",
 parameter pmi_reg_signeda_1_clk = "CLK0",
 parameter pmi_reg_signeda_1_ce = "CE0",
 parameter pmi_reg_signeda_1_rst = "RST0",
 parameter pmi_reg_signedb_0_clk = "CLK0",
 parameter pmi_reg_signedb_0_ce = "CE0",
 parameter pmi_reg_signedb_0_rst = "RST0",
 parameter pmi_reg_signedb_1_clk = "CLK0",
 parameter pmi_reg_signedb_1_ce = "CE0",
 parameter pmi_reg_signedb_1_rst = "RST0",
 parameter pmi_reg_addnsub1_0_clk = "CLK0",
 parameter pmi_reg_addnsub1_0_ce = "CE0",
 parameter pmi_reg_addnsub1_0_rst = "RST0",
 parameter pmi_reg_addnsub1_1_clk = "CLK0",
 parameter pmi_reg_addnsub1_1_ce = "CE0",
 parameter pmi_reg_addnsub1_1_rst = "RST0",
 parameter pmi_reg_addnsub3_0_clk = "CLK0",
 parameter pmi_reg_addnsub3_0_ce = "CE0",
 parameter pmi_reg_addnsub3_0_rst = "RST0",
 parameter pmi_reg_addnsub3_1_clk = "CLK0",
 parameter pmi_reg_addnsub3_1_ce = "CE0",
 parameter pmi_reg_addnsub3_1_rst = "RST0",
 parameter module_type = "pmi_dsp_multaddsubsum")

(input [(pmi_dataa_width-1):0]       A0, A1, A2, A3,
 input [(pmi_datab_width-1):0]       B0, B1, B2, B3,
 input [17:0]       SRIA,
 input [17:0]       SRIB,
 input CLK0,
 input CLK1,
 input CLK2,
 input CLK3,
 input CE0,
 input CE1,
 input CE2,
 input CE3,
 input RST0,
 input RST1,
 input RST2,
 input RST3,
 input SignA,
 input SignB,
 input ShiftA0,
 input ShiftA1,
 input ShiftA2,
```

```
    input ShiftA3,
    input ShiftB0,
    input ShiftB1,
    input ShiftB2,
    input ShiftB3,
    input ADDNSUB1,
    input ADDNSUB3,
    output [(pmi_dataa_width + pmi_datab_width + 1):0]  SUM,
    output [17:0]  SROA,
    output [17:0]  SROB)/*synthesis syn_black_box */;

endmodule //pmi_dsp_multaddsubsum
```

VHDL pmi_dsp_multaddsubsum Definition

```
component pmi_dsp_multaddsubsum is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       pmi_additional_pipeline : integer := 1;
       pmi_input_reg : string := "on";
       pmi_output_reg : string := "on";
       pmi_family : string := "ECP2";
       pmi_gsr : string := "disable";
       pmi_source_control_a0 : string := "parallel";
       pmi_source_control_a1 : string := "parallel";
       pmi_source_control_a2 : string := "parallel";
       pmi_source_control_a3 : string := "parallel";
       pmi_source_control_b0 : string := "parallel";
       pmi_source_control_b1 : string := "parallel";
       pmi_source_control_b2 : string := "parallel";
       pmi_source_control_b3 : string := "parallel";
       pmi_reg_inputa0_clk : string := "CLK0";
       pmi_reg_inputa0_ce : string := "CE0";
       pmi_reg_inputa0_rst : string := "RST0";
       pmi_reg_inputa1_clk : string := "CLK0";
       pmi_reg_inputa1_ce : string := "CE0";
       pmi_reg_inputa1_rst : string := "RST0";
       pmi_reg_inputa2_clk : string := "CLK0";
       pmi_reg_inputa2_ce : string := "CE0";
       pmi_reg_inputa2_rst : string := "RST0";
       pmi_reg_inputa3_clk : string := "CLK0";
       pmi_reg_inputa3_ce : string := "CE0";
       pmi_reg_inputa3_rst : string := "RST0";
       pmi_reg_inputb0_clk : string := "CLK0";
       pmi_reg_inputb0_ce : string := "CE0";
       pmi_reg_inputb0_rst : string := "RST0";
       pmi_reg_inputb1_clk : string := "CLK0";
       pmi_reg_inputb1_ce : string := "CE0";
       pmi_reg_inputb1_rst : string := "RST0";
       pmi_reg_inputb2_clk : string := "CLK0";
       pmi_reg_inputb2_ce : string := "CE0";
       pmi_reg_inputb2_rst : string := "RST0";
       pmi_reg_inputb3_clk : string := "CLK0";
       pmi_reg_inputb3_ce: string := "CE0";
       pmi_reg_inputb3_rst : string := "RST0";
       pmi_reg_pipeline0_clk : string := "CLK0";
       pmi_reg_pipeline0_ce : string := "CE0";
       pmi_reg_pipeline0_rst : string := "RST0";
```

```
        pmi_reg_pipeline1_clk : string := "CLK0";
        pmi_reg_pipeline1_ce : string := "CE0";
        pmi_reg_pipeline1_rst : string := "RST0";
        pmi_reg_pipeline2_clk : string := "CLK0";
        pmi_reg_pipeline2_ce : string := "CE0";
        pmi_reg_pipeline2_rst : string := "RST0";
        pmi_reg_pipeline3_clk : string := "CLK0";
        pmi_reg_pipeline3_ce : string := "CE0";
        pmi_reg_pipeline3_rst : string := "RST0";
        pmi_reg_output_clk : string := "CLK0";
        pmi_reg_output_ce : string := "CE0";
        pmi_reg_output_rst : string := "RST0";
        pmi_reg_signeda_0_clk : string := "CLK0";
        pmi_reg_signeda_0_ce : string := "CE0";
        pmi_reg_signeda_0_rst : string := "RST0";
        pmi_reg_signeda_1_clk : string := "CLK0";
        pmi_reg_signeda_1_ce : string := "CE0";
        pmi_reg_signeda_1_rst : string := "RST0";
        pmi_reg_signedb_0_clk : string := "CLK0";
        pmi_reg_signedb_0_ce : string := "CE0";
        pmi_reg_signedb_0_rst : string := "RST0";
        pmi_reg_signedb_1_clk : string := "CLK0";
        pmi_reg_signedb_1_ce : string := "CE0";
        pmi_reg_signedb_1_rst : string := "RST0";
        pmi_reg_addnsub1_0_clk : string := "CLK0";
        pmi_reg_addnsub1_0_ce : string := "CE0";
        pmi_reg_addnsub1_0_rst : string := "RST0";
        pmi_reg_addnsub1_1_clk: string := "CLK0";
        pmi_reg_addnsub1_1_ce : string := "CE0";
        pmi_reg_addnsub1_1_rst : string := "RST0";
        pmi_reg_addnsub3_0_clk : string := "CLK0";
        pmi_reg_addnsub3_0_ce : string := "CE0";
        pmi_reg_addnsub3_0_rst : string := "RST0";
        pmi_reg_addnsub3_1_clk : string := "CLK0";
        pmi_reg_addnsub3_1_ce : string := "CE0";
        pmi_reg_addnsub3_1_rst : string := "RST0";
        module_type : string := "pmi_dsp_multaddsubsum"
    );
    port (
     A0, A1, A2, A3 : in std_logic_vector((pmi_dataa_width-1)
downto 0);
     B0, B1, B2, B3 : in std_logic_vector((pmi_datab_width-1)
downto 0);
     SRIA : in std_logic_vector(17 downto 0);
     SRIB : in std_logic_vector(17 downto 0);
     CLK0: in std_logic;
     CLK1: in std_logic;
     CLK2: in std_logic;
     CLK3: in std_logic;
     CE0: in std_logic;
     CE1: in std_logic;
     CE2: in std_logic;
     CE3: in std_logic;
     RST0: in std_logic;
     RST1: in std_logic;
     RST2: in std_logic;
     RST3: in std_logic;
     SignA: in std_logic;
     SignB: in std_logic;
```

```
        ShiftA0: in std_logic;
        ShiftA1: in std_logic;
        ShiftA2: in std_logic;
        ShiftA3: in std_logic;
        ShiftB0: in std_logic;
        ShiftB1: in std_logic;
        ShiftB2: in std_logic;
        ShiftB3: in std_logic;
        ADDNSUB1: in std_logic;
        ADDNSUB3: in std_logic;
        SUM : out std_logic_vector((pmi_dataa_width +
pmi_datab_width + 1) downto 0);
        SROA : out std_logic_vector(17 downto 0);
        SROB : out std_logic_vector(17 downto 0)
      );
    end component pmi_dsp_multaddsubsum;
```

Parameter Names and Values

The following table contains the pmi_dsp_multaddsubsum module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 72	8	
pmi_datab_width	2 to 72	8	
pmi_additional_pipeline	0 to 1	0	
pmi_input_reg	“on”	“off”	“on”	
pmi_output_reg	“on”	“off”	“on”	
pmi_family	“ECP”	“ECP2”	“ECP2M”	“ECP3”	“ECP5U”	“ECP5UM”	“XP2”	“ECP2”	
pmi_gsr	“enable”	“disable”	“enable”	
pmi_source_controla0	“parallel”	“shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_source_controla1	“parallel” | “shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controla2	“parallel” | “shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controla3	“parallel” | “shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controlb0	“parallel” | “shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controlb1	“parallel” | “shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_source_controlb2	“parallel” | “shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_source_controlb3	“parallel” | “shift”	“parallel”	The pmi_source_controla0, pmi_source_controla1, pmi_source_controla2, pmi_source_controla3, pmi_source_controlb0, pmi_source_controlb1, pmi_source_controlb2 and pmi_source_controlb3 parameters have options parallel/shift for LatticeECP. This option is for LatticeECP only.
pmi_reg_inputa0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputa0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputa0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputa1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputa1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputa1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputa2_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputa2_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputa2_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputa3_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputa3_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputa3_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputb0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputb0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_reg_inputb0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputb1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputb1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputb1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputb2_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputb2_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputb2_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_inputb3_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_inputb3_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_inputb3_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_pipeline0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_pipeline0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_pipeline0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_pipeline1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_pipeline1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_pipeline1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_pipeline2_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_pipeline2_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_pipeline2_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_pipeline3_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	
pmi_reg_pipeline3_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”	
pmi_reg_pipeline3_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”	
pmi_reg_output_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_reg_output_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_output_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signeda_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signeda_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signeda_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signeda_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signeda_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signeda_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signedb_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signedb_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signedb_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_signedb_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_signedb_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_signedb_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub1_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub1_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_addnsub1_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_addnsub1_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub1_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_addnsub1_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	
pmi_reg_addnsub3_0_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub3_0_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_addnsub3_0_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_reg_addnsub3_1_rst	“RST0”	“RST1”	“RST2”	“RST3”	“RST0”	
pmi_reg_addnsub3_1_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”	
pmi_reg_addnsub3_1_ce	“CE0”	“CE1”	“CE2”	“CE3”	“CE0”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

Multiplier

Multiplier. Look-up table (LUT) implementation available. For DSP devices, DSP implementation is also available.

The signed or unsigned, n x m-bit multiplier module is an n and m input / n + m output device that performs electronically in a manner that can be interpreted as the binary multiplication of an n-bit word with the other m-bit word. The constant coefficient option allows you to specify a constant integer value for the multiplicand. A pipeline option introduces registered elements controlled by clock, clock enable, and reset ports.

Table 225: Multiplier Dialog Box

Feature	Description
Use a Constant Coefficient	Allows the operation of a constant multiplier when selected.
Constant Coefficient Value	The constant coefficient value range is -2^31 to 2^31-1. The default value is set to 2.
Use RAM Based Multiplier	Specifies RAM-based implementation of multiplier. Option only available when using Constant Coefficient feature.
Block Implementation	Allows you to choose between LUT or DSP Based Implementation.
Input A Width	Size of multiplier input. The size of the multiplier input can be any size between 2 and 36 bits. The default size is 9.
Input B Width	Size of multiplicand input. The size of the multiplicand input can be any size between 2 and 36 bits. The default size is 9.
Representation	Allows the operation of signed and unsigned multiplication. Default setting is signed.
Product Bit Width	Size of Product. The product bit width is the sum of the multiplier and multiplicand bit widths. The default size is 18.
Specify the Number of Pipeline Stages	Specifies the number of register stages inserted into the multiplier. More stages improves achievable frequency of operation, but the latency for the output increases.
Enable Input Registers	Allows the implementation of registers on the inputs. Default is input register is enabled.

Table 225: Multiplier Dialog Box (Continued)

Feature	Description
Enable Output Registers	Allows the implementation of registers on the outputs. Default is output register is enabled.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 226: Multiplier Ports

Name	Description
DataA[size-1..0]	Multiplicand input
DataB[size-1..0] (Optional)	Multiplier input
Result[size-1..0]	Multiplier output
Clock (Optional)	Positive, edge-triggered clock input for pipeline stages
ClkEn (Optional)	Clock enable input
Aclr (Optional)	Asynchronous clear input for pipeline stages

Table 227: Multiplier Properties

Feature	Description
Constant coefficient	Treat the multiplicand as an integer constant
Size	Width of the DataA, DataB, and Result ports.
Representation	Type of multiplication performed: Signed or unsigned.
Number of pipeline stages	Number of register stages inserted in the multiplier. For example, in an 8x8 multiplier: The value 1 inserts registers in one of the multiplication stages. The value 2 inserts registers in two of the multiplication stages, and so on for 4 and 8 values. Zero (0) indicates that the module should be implemented as a purely combinatorial function.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195

pmi_mult

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_mult.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA	Bus	(pmi_dataa_width - 1):0
I	DataB	Bus	(pmi_datab_width - 1:):0
I	Clock	Bit	N/A
I	ClkEn	Bit	N/A
I	Aclr	Bit	N/A
O	Result	Bus	(pmi_dataa_width + pmi_datab_width - 1):0

Verilog pmi_mult Definition

```
module pmi_mult #(parameter pmi_dataa_width = 8,
         parameter pmi_datab_width = 8,
         parameter module_type = "pmi_mult",
         parameter pmi_sign = "on",
         parameter pmi_additional_pipeline = 1,
         parameter pmi_input_reg = "on",
         parameter pmi_output_reg = "on",
         parameter pmi_family = "EC",
         parameter pmi_implementation = "LUT")

        (input   [(pmi_dataa_width-1):0]  DataA,
         input   [(pmi_datab_width-1):0]  DataB,
         input   Clock, ClkEn, Aclr,
         output  [(pmi_dataa_width + pmi_datab_width - 1):0]
Result)/*synthesis syn_black_box*/;

endmodule // pmi_mult
```

VHDL pmi_mult Definition

```
component pmi_mult is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       module_type : string := "pmi_mult";
       pmi_sign : string := "on";
       pmi_additional_pipeline : integer := 1;
       pmi_input_reg : string := "on";
       pmi_output_reg : string := "on";
       pmi_family : string := "EC";
       pmi_implementation : string := "LUT"
    );
    port (
     DataA : in std_logic_vector((pmi_dataa_width-1) downto 0);
     DataB : in std_logic_vector((pmi_datab_width-1) downto 0);
```

```
     Clock: in std_logic;
     Result : out std_logic_vector((pmi_dataa_width +
pmi_datab_width - 1) downto 0)
   );
  end component pmi_mult;
```

Parameter Names and Values

The following table contains the pmi_mult module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 36	8	
pmi_datab_width	2 to 36	8	
pmi_additional_pipeline	0 to Maximum {pmi_dataa_width, pmi_datab_width}	1	The pmi_additional_pipeline parameter ranges from 0 to 1 for DSP implementation. For the LUT implementation the limit is set to Maximum {pmi_dataa_width, pmi_datab_width}.
pmi_input_reg	“on” | “off”	“on”	
pmi_output_reg	“on” | “off”	“on”	
pmi_sign	“on” | “off”	“on”	
pmi_family	“EC” | “XP” | “XP2” | “SC” | “SCM” | “ECP” | “ECP2” | “ECP2M” | “XO” | “XO2” | “XO3L” | “ECP3” | “ECP5U” | “ECP5UM” | “LPTM”	“EC”	
pmi_implementation	“DSP” | “LUT”	“LUT”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

pmi_constant_mult

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_constant_mult.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA	Bus	(pmi_dataa_width - 1):0
I	Clock	Bit	N/A
I	ClkEn	Bit	N/A
I	Aclr	Bit	N/A
O	Result	Bus	(pmi_dataa_width + pmi_datab_width - 1):0

Verilog pmi_constant_mult Definition

```
module pmi_constant_mult #(parameter pmi_dataa_width = 8,
                 parameter pmi_datab_width = 8,
                 parameter module_type = "pmi_constant_mult",
                 parameter pmi_sign = "on",
                 parameter pmi_additional_pipeline = 1,
                 parameter pmi_constant_value = 2,
                 parameter pmi_input_reg = "on",
                 parameter pmi_output_reg = "on",
                 parameter pmi_family = "EC",
                 parameter pmi_implementation = "LUT")

              (input   [(pmi_dataa_width-1):0]  DataA,
               input   Clock, ClkEn, Aclr,
               output  [(pmi_dataa_width + pmi_datab_width -
1):0]  Result)/*synthesis syn_black_box*/;

endmodule // pmi_constant_mult
```

VHDL pmi_constant_mult Definition

```
component pmi_constant_mult is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       module_type : string := "pmi_constant_mult";
       pmi_sign : string := "on";
       pmi_additional_pipeline : integer := 1;
       pmi_constant_value : integer := 2;
       pmi_input_reg : string := "on";
       pmi_output_reg : string := "on";
       pmi_family : string := "EC";
       pmi_implementation : string := "LUT"
    );
    port (
     DataA : in std_logic_vector((pmi_dataa_width-1) downto 0);
     Clock: in std_logic;
     Result : out std_logic_vector((pmi_dataa_width +
pmi_datab_width - 1) downto 0)
   );
  end component pmi_constant_mult;
```

Parameter Names and Values

The following table contains the pmi_constant_mult module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 36	8	
pmi_datab_width	2 to 36	8	The pmi_datab_width parameter is implemented to calculate the data width for the constant value.
pmi_sign	“on” | “off”	“on”	
pmi_additional_pipeline	0 to Maximum {pmi_dataa_width, pmi_datab_width}	1	The pmi_additional_pipeline parameter ranges from 0 to 1 for DSP implementation. For the LUT implementation the limit is set to Maximum {pmi_dataa_width, pmi_datab_width}.
pmi_constant_value	-2^{31} to 2^{31} -1	2	
pmi_input_reg	“on” | “off”	“on”	
pmi_output_reg	“on” | “off”	“on”	
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” | “LPTM”	“EC”	
pmi_implementation	“DSP” | “LUT”	“LUT”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

pmi_complex_mult

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_complex_mult.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA_Re	Bus	(pmi_dataa_width - 1):0
I	DataA_Im	Bus	(pmi_dataa_width - 1):0
I	DataB_Re	Bus	(pmi_dataa_width - 1):0
I	DataB_Im	Bus	(pmi_dataa_width - 1):0
I	Clock	Bit	N/A
I	ClkEn	Bit	N/A
I	Aclr	Bit	N/A
O	Result_Re	Bus	(pmi_dataa_width + pmi_datab_width):0
O	Result_Im	Bus	(pmi_dataa_width + pmi_datab_width):0

Verilog pmi_complex_mult Definition

```
module pmi_complex_mult #(parameter pmi_dataa_width = 8,
         parameter pmi_datab_width = 8,
         parameter module_type = "pmi_complex_mult",
         parameter pmi_sign = "on",
         parameter pmi_additional_pipeline = 1,
         parameter pmi_input_reg = "on",
         parameter pmi_output_reg = "on",
         parameter pmi_family = "EC",
         parameter pmi_mult_mode = 3,
         parameter pmi_implementation = "LUT")

        (input   [(pmi_dataa_width-1):0]  DataA_Re,
         input   [(pmi_dataa_width-1):0]  DataA_Im,
         input   [(pmi_datab_width-1):0]  DataB_Re,
         input   [(pmi_datab_width-1):0]  DataB_Im,
         input   Clock, ClkEn, Aclr,
         output  [(pmi_dataa_width + pmi_datab_width):0]
Result_Re,
         output  [(pmi_dataa_width + pmi_datab_width):0]
Result_Im)/*synthesis syn_black_box*/;

endmodule // pmi_complex_mult
```

VHDL pmi_complex_mult Definition

```
component pmi_complex_mult is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       module_type : string := "pmi_complex_mult";
       pmi_sign : string := "on";
       pmi_additional_pipeline : integer := 1;
       pmi_input_reg : string := "on";
       pmi_output_reg : string := "on";
```

```
        pmi_family : string := "EC";
        pmi_mult_mode : integer := 3;
        pmi_implementation : string := "LUT"
    );
    port (
     DataA_Re : in std_logic_vector((pmi_dataa_width-1) downto
0);
     DataA_Im : in std_logic_vector((pmi_dataa_width-1) downto
0);
     DataB_Re : in std_logic_vector((pmi_datab_width-1) downto
0);
     DataB_Im : in std_logic_vector((pmi_datab_width-1) downto
0);
     Clock, ClkEn, Aclr : in std_logic;
     Result_Re : out std_logic_vector((pmi_dataa_width +
pmi_datab_width) downto 0);
     Result_Im : out std_logic_vector((pmi_dataa_width +
pmi_datab_width) downto 0)
   );
  end component pmi_complex_mult;
```

Parameter Names and Values

The following table contains the pmi_complex_mult module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 36	8	
pmi_datab_width	2 to 36	8	The pmi_datab_width parameter is implemented to calculate the data width for the constant value.
pmi_sign	“on” | “off”	“on”	
pmi_additional_pipeline	0 to Maximum {pmi_dataa_width, pmi_datab_width}	1	The pmi_additional_pipeline parameter ranges from 0 to 1 for DSP implementation. For the LUT implementation the limit is set to Maximum {pmi_dataa_width, pmi_datab_width}.
pmi_input_reg	“on” | “off”	“on”	
pmi_output_reg	“on” | “off”	“on”	
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” | “LPTM”	“EC”	
pmi_mult_mode	3 to 4	3	
pmi_implementation	“DSP” | “LUT”	“LUT”	

See Also

- "Use PMI or IPexpress?" on page 191
- "Using PMI" on page 200

Multiply_Accumulate

A two-input multiply/accumulate (MAC) logic. Look-up table (LUT) implementation is available. For DSP devices, DSP implementation is also available.

Table 228: Multiply_Accumulate Dialog Box

Feature	Description
Block Implementation	Allows you to choose between LUT based implementation and DSP. DSP is only available with ECP5, LatticeECP, LatticeECP2/M, LatticeECP3, and LatticeXP2.
Add/Sub Operation	Allows the implementation of adder or subtractor. Default is set to add.
Input A Width	Size of multiplier input. The size of the multiplier input can be any size between 2 and 36 bits. The default size is 18.
Input B Width	Size of multiplicand input. The size of the multiplicand input can be any size between 2 and 36 bits. The default size is 18.
Representation	Allows the operation of signed and unsigned multiplication. Default setting is signed.
Accumulator Width	Size of the accumulator. The accumulator can be set to any size between 1 and 32 bits. The default size is 16.
Sum Output Bit Width	Sum of the multiplier, multiplicand, and accumulator width.
Specify the Number of Pipeline Stages	Allows the number of pipelined stages to be set. Default is set to 1.
Enable Input Registers	Allows the implementation of registers on the inputs. Default is input register is enabled.
Enable Output Registers	Allows the implementation of registers on the outputs. For the Multiply Accumulate module, the output register is always enabled.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195

pmi_mac

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_mac.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA	Bus	(pmi_dataa_width - 1):0
I	DataB	Bus	(pmi_datab_width - 1):0
I	Clock	Bit	N/A
I	ClkEn	Bit	N/A
I	Aclr	Bit	N/A
O	Result	Bus	(pmi_accum_width - 1):0

Verilog pmi_mac Definition

```
module pmi_mac #(parameter pmi_dataa_width = 8,
         parameter pmi_datab_width = 8,
         parameter pmi_accum_width = 32,
         parameter module_type = "pmi_mac",
         parameter pmi_sign = "on",
         parameter pmi_additional_pipeline = 1,
         parameter pmi_add_sub = "add",
         parameter pmi_input_reg = "on",
         parameter pmi_family = "EC",
         parameter pmi_implementation = "LUT")

        (input   [(pmi_dataa_width - 1):0]  DataA,
         input   [(pmi_datab_width - 1):0]  DataB,
         input   Clock, ClkEn, Aclr,
         output  [(pmi_accum_width - 1):0]  Result)/*synthesis
syn_black_box*/;

endmodule // pmi_mac
```

VHDL pmi_mac Definition

```
component pmi_mac is
     generic (
       pmi_dataa_width : integer := 8;
       pmi_datab_width : integer := 8;
       pmi_accum_width : integer := 32;
       module_type : string := "pmi_mac";
       pmi_sign : string := "on";
       pmi_additional_pipeline : integer := 1;
       pmi_add_sub : string := "add";
       pmi_input_reg : string := "on";
       pmi_family : string := "EC";
       pmi_implementation : string := "LUT"
    );
    port (
```

```
      DataA : in std_logic_vector((pmi_dataa_width - 1) downto
0);
      DataB : in std_logic_vector((pmi_datab_width - 1) downto
0);
      Clock: in std_logic;
      Result : out std_logic_vector((pmi_accum_width - 1) downto
0)
    );
  end component pmi_mac;
```

Parameter Names and Values

The following table contains the pmi_mac module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_dataa_width	2 to 36	8	
pmi_datab_width	2 to 36	8	
pmi_sign	“on” | “off”	“on”	
pmi_accum_width	(pmi_dataa_width + pmi_datab_width + 1) to (pmi_dataa_width + pmi_datab_width + 32)	“32”	
pmi_additional_pipeline	0 to Maximum {pmi_dataa_width, pmi_datab_width}	1	The pmi_additional_pipeline parameter ranges from 0 to 1 for DSP implementation. For the LUT implementation the limit is set to Maximum {pmi_dataa_width, pmi_datab_width}.
pmi_add_sub	“add” | “sub”	“add”	
pmi_input_reg	“on” | “off”	“on”	
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO2L” |“LPTM”	“EC”	
pmi_implementation	“DSP” | “LUT”	“LUT”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

ORCAstra

The interface for the ORCAstra FPGA Bus Control Center (not available with Diamond). The module includes a JTAG port for external communication. For LatticeSC/M, internal communication is with the User Master Interface of a System Bus module. For LatticeECP2M, LatticeECP3, and LatticeXP2, internal communication is with a SERDES Client Interface (SCI).

Note

The ORCAstra module cannot be simulated. The usual practice is to add the module when your design is ready for on-chip debugging.

Also, you must use Synplify to synthesize the design when the ORCAstra module is included.

Table 229: ORCAstra Dialog Box

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Ports The JTAG port is automatically connected to the JTAG pins, so the JTAG signals are not discussed here or shown in the ORCAstra Configuration dialog box. No user action is required.

Table 230:

Name	Description
clk	Free-running clock input. It is typically connected to the output of an OSCA clock library element but any stable clock will suffice. The frequency should be specified in the constraints file during place-and-route.
jcex	The scien port in the SCI bus.
rstn	Active-low, asynchronous reset. Its use is optional because the logic is self-starting and will start from any initial state.
sci<*name*>	All port names starting with “sci” are from an SCI bus and connect with an address decoder of your own design. See “References,” below.
umi_<*name*>	All port names starting with “umi_” connect with the User Master Interface of a LatticeSC/M System Bus module. See “References,” below.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- TN1085, *LatticeSC MPI/System Bus*

- TN1124 - *LatticeECP2M SERDES/PCS Usage Guide*

PCS

PCS (Physical Coding Sub-layer) supports advanced system level standards including RapidIO, Sonet, Gigabit Ethernet, 10 Gb Ethernet, Fibre Channel and PCI Express.

In addition to the usual files, the PCS module includes a <*module_name*>.txt file. This file is used by simulation and bitgen to configure the modules. When you import the source files for this module into the design project, also import the <*module_name*>.txt file as a source. It will show up under Documents in the Files tab.

The options vary with the device family. See:

- "PCS Options (ECP5UM)" on page 1556
- "PCS Options (LatticeECP2M)" on page 1560
- "PCS Options (LatticeECP3)" on page 1565
- "PCS Options (LatticeSC/M)" on page 1571

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with PCS/SERDES" on page 1166
- TN1261, ECP5 SERDES/PCS Usage Guide
- TN1124 - *LatticeECP2M SERDES/PCS Usage Guide*

PCS Options (ECP5UM)

PCS for ECP5UM is available through System Builder/Planner, not IPexpress.

The dialog box has several tabs:

- Instance Setup, see Table 231 on page 1557
- SerDes Setup, see Table 232 on page 1558
- PCS Setup, see Table 233 on page 1558
- Control Setup, see Table 234 on page 1559
- Advanced Setup, see Table 235 on page 1560

Table 231: PCS Dialog Box for ECP5UM—Instance Setup Tab

Feature	Description
Protocol	Specify the communication protocol.
Number of Channels	Specify the number of channels.
Mode	Specify if the channels are to both receive (RX) and transmit (TX), transmit only, or receive only.
Tx Max Data Rate	Specify the maximum transmit data rate in giga-bits per second (Gbps).
PLL Multiplier	Select a clock multiplier to achieve the transmit data rate. This value determines the required reference clock frequency (below).
Ref Clk Freq	Shows the reference clock frequency required to achieve the transmit data rate given the selected PLL multiplier (in MHz). Adjust the Tx Max Data Rate and the PLL Multiplier (above) until you are satisfied with the reference clock frequency.
Loss Of Lock Setting	Allowed loss of signal threshold selections for a selected protocol.
Receive Max Data Rate (CDR)	Specify the maximum receive data rate in giga-bits per second (Gbps).
CDR Multiplier	Select a clock multiplier to achieve the receive data rate. This value determines the required reference clock frequency (below).
Rx Rate	Specify if the receive rate is full speed or 1/2 of the Transmit Data Rate (above).
CDR Refclk	Shows the reference clock frequency required to achieve the receive data rate given the selected CDR multiplier (in MHz). Adjust the Receive Max Data Rate and the CDR Multiplier (above) until you are satisfied with the reference clock frequency.
Rx Line Rate	Shows the receive line rate based on the Receive Max Data Rate and the Rx Rate (above).
Rx FPGA Bus Width	Select the desired receive bus width.
Rx FPGA Bus Freq	Shows the receive bus frequency based on the Receive Max Data Rate and the Rx Bus Width (above).
Rx Low Data Rate Path	Select if the channel is to be low-speed.
CDR Loss Of Lock Range	Allowed loss of signal threshold selections for a selected protocol.
Tx Rate	Specify if the transmit rate is full speed or 1/2 of the Transmit Data Rate (above).
Tx Line Rate	Shows the transmit line rate based on the Tx Max Data Rate and the Tx Rate (above).
Tx FPGA Bus Width	Select the desired transmit bus width.
Tx FPGA Bus Freq	Shows the receive bus frequency based on the Tx Max Data Rate and the Tx Bus Width (above).
Tx Low Data Rate Path	Select if the channel is to be low-speed.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.

Table 231: PCS Dialog Box for ECP5UM—Instance Setup Tab (Continued)

Feature	Description
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 232: PCS Dialog Box for ECP5UM—SerDes Setup Tab

Feature	Description
I/O Standard	Select a SerDes standard.
Differential Amplitude	Select an amplitude level.
Output Termination	Select the amount of resistance for the termination.
De-emphasis Pre-cursor Select	Select a value or disable it.
De-emphasis Post-cursor Select	Select a value or disable it.
Input Termination	Select the amount of resistance for the termination.
AC/DC Couple	Choose either AC or DC coupling.
Linear Equalizer	Select a value or disable it.
Loss of Signal Threshold Select	Select a loss-of-signal threshold.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 233: PCS Dialog Box for ECP5UM—PCS Setup Tab

Feature	Description
Invert Tx Data Polarity	Choose whether to invert the signal. When enabled, the P data goes on the N output and the N data goes on the P output.
Enable 8B10B Encoder	Choose whether to use the 8b10b encoder.
FPGA Bridge Tx FIFO	Specify if you want a FIFO bridge.
Invert Rx Data Polarity	Choose whether to invert the signal. When enabled, the P data is expected on the N input and the N data is expected on the P input.
Enable 8B10B Decoder	Choose whether to use the 8b10b decoder.
FPGA Bridge Rx FIFO	Specify if you want a FIFO bridge.
Enable External Link State Machine	Adds External Link State Machine ports to control the word aligner.
Word Alignment (WA) Mode	Select a word alignment mode.

Table 233: PCS Dialog Box for ECP5UM—PCS Setup Tab (Continued)

Feature	Description
Specific Comma	Choose from K28P5 and K28P157. The fields below show the resulting comma A, B, and mask characters.
Comma A, B, Mask Char	Shows the comma A, B, and mask characters depending on the specific comma selected above.
Enable CTC FIFO	Select to include a CTC (clock tolerance compensation) block.
Match Pattern	Choose a CTC match pattern.
Byte N	Specify the10-bit CC register value. If K is selected, the two prefix bits (MSB) are 01. If K is cleared, the prefix bits are 00. Specify the remaining eight bits in the text box.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 234: PCS Dialog Box for ECP5UM—Control Setup Tab

Feature	Description
Loopback Mode	Choose a loopback mode.
Dynamic Rate Control Interface	Choose whether or not to enable the dynamic rate control interface.
Receive Loss-of-Signal Port	Choose whether or not to enable the receive loss-of-signal port.
CDR LOL Action	Choose the kind of recalibration following a loss-of-lock: none, simple, or full.
Select Reset Sequence	Select to include a reset sequence. Then enable the Tx Ready Port, Rx Ready Port, or both, and specify the wait times.
Import	1. If you are testing the serdes with Reveal Analyzer, click **Import**. This opens the Import Reveal SERDES Analyzer Setting dialog box. 2. Click the **...** button. 3. In the Open dialog box, browse to the .srv file. 4. Click **Open**. 5. Specify the DCU and channel. 6. Click **OK**.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 235: PCS Dialog Box for ECP5UM—Advanced Setup Tab

Feature	Description
SCI Port Enable	Select to add a SerDes Client Interface (SCI) to help with debugging SerDes.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with PCS/SERDES" on page 1166
- TN1261, ECP5 SERDES/PCS Usage Guide

PCS Options (LatticeECP2M)

The dialog box has several tabs:

- Quad, see below
- Reference Clocks, see Table 237 on page 1561

  This tab is enabled if Quad-Based Protocol Mode is selected in the Quad tab.
- Reference Clocks (CM), see Table 238 on page 1562

  This tab is enabled if Channel-Based Protocol Mode is selected in the Quad tab.
- SerDes Advanced, see Table 239 on page 1562
- PCS Advanced, see Table 240 on page 1563
- Optional Port, see Table 241 on page 1564
- Generation, see Table 242 on page 1565

Table 236: PCS Dialog Box for LatticeECP2M—Quad Tab

Feature	Description
Protocol Setup	**Quad Based Protocol Mode** - Selecting this option enables the Reference Clocks tab. **Channel Based Protocol Mode** - Selecting this option enables the Reference Clocks (CM) tab. **Quad Protocol Mode** - Drop-down menu allows you to select protocol.
Channel Selection and Grouping	Controls group mode selection for Channel0, Channel1, Channel2 and Channel3. The supported group modes are: Single, MCA Group1, MCA Group2, and Disable. MCA Group 1 and 2 are only available in the Quad Based Protocol Mode and depending on the Quad Protocol Mode chosen. Setting the channel rate is only available in the Channel Based Protocol Mode.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 237: PCS Dialog Box for LatticeECP2M—Reference Clocks Tab

Feature	Description
Tx Reference Clock Source Selection	Use REFCLK for external clock source. Use CORE_TXREFCLK for internal clock source.
Rx Reference Clock Source Selection	Use REFCLK for external clock source. Use CORE_RXREFCLK for internal clock source.
Serial Bit Clock Rate	Required serial bit clock rate for application (in GHz).
Reference Clock Multiplier	Allowed PCS reference clock multiplier values for given serial bit clock rate.
Calculated Reference Clock Rate	Resulting input reference clock required for the supplied serial bit rate given the selected reference clock multiplier (in MHz). The value is calculated automatically based on the Serial Bit Clock Rate and the Reference Clock Multiplier (above).
FPGA Interface Data Bus Width	Allowed bus width selections for the selected PCS Quad protocol (single or double bus width).
Calculated FPGA Interface Clock Rate	Resulting FPGA interface clock required for the supplied serial bit rate given the selected FPGA interface data bus width. The value is calculated automatically based on the Serial Bit Clock Rate and the FPGA Interface Data Bus Width (above).
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 238: PCS Dialog Box for LatticeECP2M—Reference Clocks (CM) Tab

Feature	Description
Tx Reference Clock Source Selection	Use REFCLK for external clock source. Use CORE_TXREFCLK for internal clock source.
Rx Reference Clock Source Selection	Selections available for Channel0, Channel1, Channel2, and Channel3. Use REFCLK for external clock source. Use CORE_RXREFCLK for internal clock source.
Serial Bit Clock Rate	Required serial bit clock rate for application (in GHz). Specify the Full Rate Channel. The Half Rate Channel is calculated automatically.
Reference Clock Multiplier	Allowed PCS reference clock multiplier values for given serial bit clock rate. The Half Rate Channel is always the same as the Full Rate Channel.
Calculated Reference Clock Rate	Resulting input reference clock required for the supplied serial bit rate given the selected reference clock multiplier (in MHz). The value is calculated automatically based on the Serial Bit Clock Rate and the Reference Clock Multiplier (above).
FPGA Interface Data Bus Width	Allowed bus width selections for the selected PCS Quad protocol (single or double bus width). Specify the Full Rate Channel and the Half Rate Channel. They can have different values.
Calculated FPGA Interface Clock Rate	Resulting FPGA interface clock required for the supplied serial bit rate given the selected FPGA interface data bus width. The value is calculated automatically based on the Serial Bit Clock Rate and the FPGA Interface Data Bus Width (above).
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 239: PCS Dialog Box for LatticeECP2M—SerDes Advanced Tab

Feature	Description
Differential Output Voltage	Allowed differential output voltage selections for a selected protocol. This is a channel-level transmit setting.
PreEmphasis	Allowed pre-emphasis selections for a selected protocol. This is a channel-level transmit setting.
TX I/O Termination(Ohms)	Allowed transmit I/O termination selections for a selected protocol. This is a channel-level transmit setting.
Equalization	Allowed equalization selections for a selected protocol. This is a channel-level receive setting.
RX I/O Termination(Ohms)	Allowed receive I/O termination selections for a selected protocol. This is a channel-level receive setting.
RX I/O Coupling	Allowed receive I/O coupling selections for a selected protocol. This is a channel-level receive setting.
Loss of Signal Threshold	Allowed loss of signal threshold selections for a selected protocol. This is a quad-level receive setting.

Table 239: PCS Dialog Box for LatticeECP2M—SerDes Advanced Tab (Continued)

Feature	Description
TX PLL Reference Clock IO Termination(Ohms)	Allowed transmit PLL reference clock I/O termination selections for a selected protocol. This is a quad-level setting.
TX Reference Clock IO Coupling	Allowed transmit PLL reference clock I/O coupling selections for a selected protocol. This is a quad-level setting.
PLL Loss of Lock	Allowed PLL loss of lock selections for a selected protocol. This is a quad-level setting.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 240: PCS Dialog Box for LatticeECP2M—PCS Advanced Tab

Feature	Description
Rx_Tx 8B10B Mode	This setting is for 8B/10B decoder bypass control and is set automatically depending on the Quad Protocol Mode set in the Quad tab. In NORMAL mode the 8B/10B decoder is not bypassed and in BYPASS mode the 8B/10B decoder will be bypassed. This is a channel level setting.
TxInvert	This setting is for controlling the inversion of transmit data. In NORMAL mode, data is not inverted. In INV mode, data is inverted. This is a channel level setting.
RxInvert	This setting is for controlling the inversion of receive data. In NORMAL mode, data is not inverted. In INV mode, data is inverted. This is a channel level setting.
Plus Comma Value	Set the positive disparity 10-bit code. This value is shown as LSb:MSb (that is, with the least significant bit to the left and the most significant bit to the right). For a selected protocol the default value will be populated. This is a quad level receive setting.
Minus Comma Value	Set the negative disparity 10-bit code. This value is shown as LSb:MSb (that is, with the least significant bit to the left and the most significant bit to the right). For a selected protocol the default value will be populated. This is a quad level receive setting.
Comma Mask	Set the10-bit comma mask value. This value is shown as LSb:MSb (that is, with the least significant bit to the left and the most significant bit to the right). For a selected protocol the default value will be populated. This is a quad level receive setting.
Comma Align	Allowed comma align selections for a selected protocol. This is a quad level receive setting.
CTC	Allowed CTC (clock tolerance compensation) selections for a selected protocol. This is a channel level receive setting.
CC_MATCH	Set the10-bit CC1, CC2, CC3, or CC4 register value. This value is shown as MSb:LSb (that is, with the most significant bit to the left and the least significant bit to the right). For a selected protocol the default value will be will populated. This is a quad level receive setting.

Table 240: PCS Dialog Box for LatticeECP2M—PCS Advanced Tab (Continued)

Feature	Description
CC_MATCH_MODE	Allowed cc match mode selections for a selected protocol. This is a quad level receive setting.
RX CTC Min IPG	Allowed CTC IPG (Inter-Packet Gap) selections for a selected protocol. This is a quad level receive setting.
High Watermark	Allowed high watermark setting on the CTC FIFO (Rx Elastic Buffer) selections for a selected protocol. This is a quad level receive setting.
Low Watermark	Allowed low watermark setting on the CTC FIFO (Rx Elastic Buffer) selections for a selected protocol. This is a quad level receive setting.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 241: PCS Dialog Box for LatticeECP2M—Optional Port Tab

Feature	Description
Dynamic Inversion of Received Data	This setting controls the inclusion of Dynamic Inversion of Received Data ports in the wrapper top-level port list. Select this option to include these ports. **Note:** This setting is different from the “Rx Invert” setting in the PCS Advanced Tab.
External Link State Machine	Adds External Link State Machine ports in the wrapper top-level port list.
Transmitter Electrical Idle	This setting controls the inclusion of Transmitter Electrical Idle ports in the wrapper top-level port list. Select this option to include these ports. Available only when the Quad Protocol Mode is Generic 8B10B.
Loopback Mode (Rx to Tx)	This setting controls Loopback options for selection.
Serial Loopback (Rx Equalization to Tx, No Clock)	This setting controls Serial Loopback selection.
SerDes Parallel Loopback	The channel-specific SerDes Parallel Loopback will be included in the wrapper top-level port list.
PCS Parallel Loopback	The channel-specific PCS Parallel Loopback will be included in the wrapper top-level port list.
Reference Clock to FPGA Core	The Reference Clock to FPGA Core Port will be included in the wrapper top-level port list.
PLL Quarter Clock	The PLL Quarter Clock Port will be included in the wrapper top-level port list.
SCI (SerDes Client Interface)	The SCI Ports will be included in the wrapper top-level port list.
SCI Interrupt	The SCI Interrupt Port will be included in the wrapper top-level port list.
Error Status	The Error Status Ports will be included in the wrapper top-level port list.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.

Table 241: PCS Dialog Box for LatticeECP2M—Optional Port Tab (Continued)

Feature	Description
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 242: PCS Dialog Box for LatticeECP2M—Generation Tab

Feature	Description
Automatic	Automatically generates the HDL and attributes (.txt) files as needed. Some changes do not require regenerating both files. **Force Place & Route Process Reset** - Select to reset the Place & Route Design process. **Force Place & Route Trace Process Reset** - Select to reset the Place & Route Trace process.
Force Module and Settings Generation	Generates both the HDL and attributes files.
Force Settings Generation Only (error if module generation is required)	Generates only the attributes file. You get an error message if the HDL file also needs to be generated. **Force Place & Route Process Reset** - Select to reset the Place & Route Design process. **Force Place & Route Trace Process Reset** - Select to reset the Place & Route Trace process.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with PCS/SERDES” on page 1166
- TN1124 - *LatticeECP2M SERDES/PCS Usage Guide*

PCS Options (LatticeECP3)

The dialog box has several tabs:

- Quad, see below
- Reference Clocks, see Table 244 on page 1567

- SerDes Advanced, see Table 245 on page 1568
- PCS Advanced1, see Table 246 on page 1568
- PCS Advanced2, see Table 247 on page 1569
- Control Setup, see Table 248 on page 1570
- Generation Options, see Table 249 on page 1570

Table 243: PCS Dialog Box for LatticeECP3—Quad Tab

Feature	Description
RX and TX, RX only, TX only, Disable Channel	Specify if the channel is to both receive (RX) and transmit (TX), receive only, transmit only, or be disabled.
Protocol	Specify the channel's communication protocol.
Low Speed Data Ports	Specify if the channel is to be low-speed.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 244: PCS Dialog Box for LatticeECP3—Reference Clocks Tab

Feature	Description
Transmit	**Max Data Rate** - Specify the transmit data rate for all channels in Gbps. The range is 0.25 - 3.2. **Tx Refclk Source** - Specify if the transmit reference clock is internal or external to the FPGA. **Tx/Rx Multiplier** - Select a multiplier for the transmit reference clock rate (see next item) and the receive reference clock rate (see the Receive row following). **Tx Reference Clock Rate** - Shows the required input reference clock rate. The rate is the Transmit Data Rate divided by the Tx/Rx Multiplier. In most cases, you can use the calculated value. **Protocol** - Shows the receive/transmit choices made in the Quad tab. **Tx Rate** - Specify if the transmit rate for this channel is full speed, or 1/2 or 1/11 of the Transmit Data Rate (above). **FPGA Bus Width** - Specify the width of the parallel bus to be transmitted on this channel. **FPGA Interface Clock** - Shows the required FPGA interface clock rate. The rate depends on the Transmit Data Rate and the transmit FPGA Bus Width of the channel.
Receive	**Max Data Rate** - Specify the receive data rate in Gbps. The range is 0.25 - 3.2. **Protocol** - Shows the receive/transmit choices made in the Quad tab. **Reference Clock Source** - Specify if the receive reference clock for this channel is internal or external to the FPGA. **Multiplier** - Shows the multiplier for the Rx reference clock rate (see below). The multiplier is selected in Tx/Rx Multiplier (see the Transmit row above). **Rx Rate** - Specify if the receive rate for this channel is full speed, or 1/2 or 1/11 of the Transmit Data Rate (above). **Rx Reference Clock Rate** - Shows the required input reference clock rate. The rate is the same as Tx Reference Clock Rate, which is the Transmit Data Rate divided by the Tx/Rx Multiplier (see the Transmit row above). **FPGA Bus Width** - Specify the width of the parallel bus to be transmitted from this channel. **FPGA Interface Clock** - Shows the required FPGA interface clock rate. The rate depends on the Transmit Data Rate (see the Transmit row above) and the receive FPGA Bus Width of the channel.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 245: PCS Dialog Box for LatticeECP3—SerDes Advanced Tab

Feature	Description
Transmit Settings	**Differential Output Voltage** - Select the differential output voltage for this channel. **PreEmphasis** - Select the pre-emphasis setting for this channel. **TX I/O Termination (Ohms)** - Select the transmit I/O termination for this channel.
Receive Settings	**Equalization Setting** - Select the equalization setting for this channel. **RX I/O Termination (Ohms)** - Select the receive I/O termination for this channel. **RX I/O Coupling** - Select the receive I/O coupling for this channel. **Loss of Signal Threshold Setting** - Select the loss of signal threshold for this channel.
Clock and PLL Settings	**Reference Clock IO Termination (Ohms)** - Allowed transmit PLL reference clock I/O termination selections for a selected protocol. **Reference Clock IO Coupling** - Allowed transmit PLL reference clock I/O coupling selections for a selected protocol. **TX PLL Loss of Lock** - Allowed PLL loss of lock selections for a selected protocol.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 246: PCS Dialog Box for LatticeECP3—PCS Advanced1 Tab

Feature	Description
Transmit Settings	**Invert the Transmit Signal** - Choose whether to invert the signal. When enabled, the P data goes on the N output and the N data goes on the P output. **Enable 8B10B Encoder** - Choose whether to use the 8b10b encoder. **FPGA FIFO Bridge** - Specify if you want a FIFO bridge on this channel.
Receive Settings	**Invert the Receive Signal** - Choose whether to invert the signal. When enabled, the P data is expected on the N input and the N data is expected on the P input. **Dynamically Invert Port** - Specify whether to dynamically invert the receive signal. When enabled, a ffc_sb_inv_rx_ch port is added to control inverting this channel. **Enable 8B10B Decoder** - This setting is for 8b10b decoder bypass control. In NORMAL mode the 8b10b decoder is not bypassed and in BYPASS mode the 8b10b decoder will be bypassed. **FPGA FIFO Bridge** - Specify if you want a FIFO bridge on this channel.

Table 246: PCS Dialog Box for LatticeECP3—PCS Advanced1 Tab (Continued)

Feature	Description
Word Alignment	**Word Alignment Block** - Choose whether to use the word alignment block. **Internal Link State Machine** - Choose whether to use the internal link state machine. **Specific Comma** - Choose from User Defined, 1111111111 (K28.5), 1111111100 (K28.157), and DISABLED. Choose User Defined to specify the plus and minus comma value, and the comma mask (following) for this channel. **Plus Comma Value** - Specify the positive disparity 10-bit code for this channel. **Minus Comma Value** - Specify the negative disparity 10-bit code for this channel. **Comma Mask** - Specify the10-bit comma mask for this channel.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 247: PCS Dialog Box for LatticeECP3—PCS Advanced2 Tab

Feature	Description
CTC block	Choose whether to include a CTC (clock tolerance compensation) block for this channel. CTC blocks are not available with LatticeECP3-150TW.
Size of the ordered set to match (Bytes)	Choose the number of bytes to match for this channel: 1, 2, or 4.
ByteN(Hex)	Specify the10-bit CC register value for this channel. If K is selected, the two prefix bits (MSB) are 01. If K is cleared, the prefix bits are 00. Specify the remaining eight bits in the text box.
Inter packet gap to enforce	Choose the minimum inter-packet gap for this channel. The minimum number of bytes after a deletion is this value plus one times the size of the ordered set (above). For example, if you choose 3 for the gap and 2 for the ordered set, the minimum gap is (3 + 1) x 2 = 8 bytes.
Deletion Threshold	Specify the point in the FIFO at which CTC block deletes an ordered set. The deletion threshold must be larger than the insertion threshold.
Insertion Threshold	Specify the point in the FIFO at which CTC block inserts an ordered set.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 248: PCS Dialog Box for LatticeECP3—Control Setup Tab

Feature	Description
Loopback Selection	Select the desired mode for each enabled channel: ▶ Disabled ▶ Loopback serial data after equalizer ▶ Loopback serial data after transmit driver ▶ Loopback parallel data after de-serializer
Transmitter Electrical Idle	This setting controls the inclusion of Transmitter Electrical Idle ports in the wrapper top-level port list. If the check box is enabled the ports will be included in the wrapper top-level port list.
Reset Sequence Generation	The PCS module for LatticeECP3 requires a reset sequence after system reset and when a receiver is reconnected. Select **Include Reset Sequence Generation** to add logic that produces the reset sequence. Highly recommended. **Note:** If you are upgrading a PCS module for an existing design and the design provides separate logic for the reset sequence, clear **Include Reset Sequence Generation**. The regenerated module will still work with the separate reset sequence logic.
Provide SerDes Client Interface (SCI)	Select to include the SerDes Client Interface (SCI).
Provide the SerDes Client Interface (SCI) Interrupt Port	Select to include the SCI Interrupt Port.
Reference Clock to FPGA Core	Select to include the refclk2fpga output.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 249: PCS Dialog Box for LatticeECP3—Generation Options Tab

Feature	Description
Automatic (Generate Module and Settings as needed)	Automatically generates the HDL and attributes (.txt) files as needed. Some changes do not require regenerating both files. **Force Place & Route Process Reset** - Select to reset the Place & Route Design process. **Force Place & Route Trace Process Reset** - Select to reset the Place & Route Trace process.
Force Module and Settings Generation	Generates both the HDL and attributes files.

Table 249: PCS Dialog Box for LatticeECP3—Generation Options Tab (Continued)

Feature	Description
Force Settings Generation Only (error if module generation required)	Generates only the attributes file. You get an error message if the HDL file also needs to be generated. **Force Place & Route Process Reset** - Select to reset the Place & Route Design process. **Force Place & Route Trace Process Reset** - Select to reset the Place & Route Trace process.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with PCS/SERDES" on page 1166

PCS Options (LatticeSC/M)

The dialog box has several tabs:

- PCS, see below
- Setup, see Table 251 on page 1572
- Multi-Channel Alignment, see Table 252 on page 1572

  Available only when one of the Align Channels options is selected in the PCS tab.

Table 250: PCS Dialog Box for LatticeSC/M—PCS Tab

Feature	Description
Quad Mode	PCS protocol selection for one PCS quad (all four channels).
Channel 0, 1, 2, 3 Enabled	Select which channels to enable. In the autoconfig file, any channel that is enabled will be powered up. Channels can later be powered down using the run time memory map.
Align Channels	Select a static multi-channel alignment mode. This alignment mode can be later changed using the run time memory map. Go to the Multi-Channel Alignment tab to set options including multi-quad alignment.
Optional Direct Control & Status Register Access	Include control and status pins on the PCS block interface.

Table 250: PCS Dialog Box for LatticeSC/M—PCS Tab (Continued)

Feature	Description
Optional System Bus Interface	Include System BUS Interface pins on the PCS block interface. Required if System bus control of the PCS is desired.
Use internal 10G (XAUI) Link State Machine	Uses a 10G (XAUI) link state machine to control the word aligner. Available only with the Generic 8b10b Quad Mode.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 251: PCS Dialog Box for LatticeSC/M—Setup Tab

Feature	Description
Reference Clock Source Selection	Select the source of the reference clock for the Tx and Rx clocks to the SERDES. If “All Reference Clocks” is selected, set the autoconfig file to the correct default for reference clock selection in quad register 0x29. The default value written in the autoconfig file is 0x00, which is the external reference clock.
Serial Bit Clock Rate (GHz)	Required serial bit clock rate for the application (in GHz).
Reference Clock Multiplier	Multiplier for the reference clock to reach the serial bit clock rate.
Calculated Reference Clock Rate (MHz)	Required input reference clock rate (in MHz) given the serial bit clock rate and the reference clock multiplier set above.
FPGA Interface Data Bus Width	Width of the data buses for each channel (single or double bus width).
FPGA Interface Clock Rate (MHz)	Required FPGA interface clock and data bus rate given the serial bit clock rate and interface data bus width set above.
Amplitude Boost	Provides an additional 100-150 mV differential amplitude for a 20% power increase.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 252: PCS Dialog Box for LatticeSC/M—Multi-Channel Alignment Tab

Feature	Description
Multi-channel Alignment Settings	Select the channel alignment character. If only one alignment word is needed, set “A” and “B” to the same value.
Enable Multi-Quad Alignment	Select Multi-Quad Alignment. Available if “Optional Direct Control & Status Register Access,” in the PCS tab, has been selected for two or more PCS Quads.

Table 252: PCS Dialog Box for LatticeSC/M—Multi-Channel Alignment Tab (Continued)

Feature	Description
Select Multi-Quad Alignment Group	Select one of four Multi-Quad Alignment input clocks (Group 0, Group 1, Group 2, or Group 3) for all quads that require Multi-Quad alignment. All Quads to be aligned to each other must use the same Multi-Quad Alignment input clock.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with PCS/SERDES” on page 1166

PLL

A phase-locked loop (PLL) that provides a variety of sysCLOCK PLL implementations. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

Frequency preferences on the output clocks are generated automatically. After generating the PLL module, you can modify the FREQUENCY preference with keywords such as PAR_ADJ and HOLD_MARGIN. See FREQUENCY.

The options vary with the device family. See:

- “PLL Options (ECP5)” on page 1574
- “PLL Options (LatticeECP/EC, LatticeXP, MachXO)” on page 1577
- “PLL Options (LatticeECP2/M)” on page 1579
- “PLL Options (LatticeECP3)” on page 1582
- “PLL Options (LatticeSC/M)” on page 1585
- “PLL Options (LatticeXP2)” on page 1587
- “PLL Options (MachXO2, MachXO3L, Platform Manager 2)” on page 1589
- “PLL Options (LIFMD)” on page 1592

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysCLOCK PLLs and DLLs” on page 1168
- TN1263, *ECP5 sysCLOCK PLL/DLL Design and Usage Guide*

- TN1049, *LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide*
- TN1103, *LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide*
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide
- TN1098 - LatticeSC sysCLOCK PLL/DLL User's Guide
- TN1126, *LatticeXP2 sysCLOCK PLL Design and Usage Guide*
- TN1089, *MachXO sysCLOCK Design and Usage Guide*
- TN1199, *MachXO2 sysCLOCK PLL Design and Usage Guide*
- TN1282, *MachXO3L sysCLOCK PLL Design and Usage Guide*
- TN1304, *CrossLink sysCLOCK PLL/DLL Design and Usage Guide*

PLL Options (ECP5)

The module dialog box has three tabs:

- Frequency, see below
- Phase, see Table 254 on page 1576
- Optional Ports, see Table 255 on page 1576

Output frequencies depend on the input frequency, the desired output frequencies, and the tolerances. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

After setting up the Frequency tab, go to the other tabs to specify phase shifts for the output clocks and optional ports such as for dynamic phase shift, clock enable, and PLL reset.

Table 253: PLL Dialog Box for ECP5—Frequency Tab

Feature	Description
CLKI	**Frequency** – CLKI is the input reference clock for the PLL. Specify its frequency in megahertz. **Refclk Divider** – After clicking Calculate, this shows the divider value used to achieve the desired output frequency. **Enable High Bandwidth** – Select to have a higher PLL bandwidth. After clicking Calculate, the bandwidth is shown in the Info box. **PLL Reference Clock from Pin** – Select to get the reference clock directly from one of the FPGA's pins. Then choose the pin's buffer type.
CLKFB	**Feedback Mode** – Specify the feedback clock source. Choices are internal and external CLKOP, CLKOS, CLKOS2, CLKOS3, and external user clock. To use CLKOS, CLKOS2, or CLKOS3, select the clock's **Enable** option and clear the **Bypass** option. Then choose the feedback mode. If you select Bypass for CLKOP, the CLKOP, INT_OP, and UserClock options are disabled. **Feedback Divider** – The divider in the feedback path is shown after clicking Calculate.
Info	These values are shown after filling in the other fields and clicking Calculate.
CLKOP, CLKOS, CLKOS2, CLKOS3	The PLL can produce up to four different clock signals. Select **Enable** for the clocks you want to use (CLKOP is always enabled). Then use the following options to specify the form of the clocks. **Enable** – Select to add the clock to the module. CLKOP is the standard output and is always enabled. **Bypass** – The bypass option connects the output to the input, bypassing the PLL circuit. **Output Divider** – Shows the calculated divider to produce the actual frequency. **Desired Frequency** – Specify the desired output clock frequency. **Tolerance %** – Specify a tolerance for the frequency as a percentage of the requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – The actual frequency that the PLL will produce is after clicking Calculate.
Calculate	Software calculates divider settings and actual output frequencies based on CLKI and desired frequencies.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 254: PLL Dialog Box for ECP5—Phase Tab

Feature	Description
Desired Phase	Specify the desired phase shift in degrees.This option is available if the clock is enabled but the Bypass option has not been selected and the clock has not been selected in Feedback Mode.
Actual Phase	Shows the actual phase shift after clicking Calculate. Not all phase shifts are possible.
Calculate	After entering the desired phase shifts, click **Calculate** to see what actual phase shift will be implemented.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 255: PLL Dialog Box for ECP5—Optional Ports Tab

Feature	Description
Enable Clock Select	Adds a second input clock port, CLKI2, and a select port.
Dynamic Phase Ports	Adds phase controls.
Clock Enable OP, OS, OS2, OS3	Adds an enable input port for each selected clock. This option is available if the clock is enabled but the Bypass option has not been selected and the clock has not been selected in Feedback Mode. The PHASESEL port selects a clock with the following code: CLKOP: 11 CLKOS: 00 CLKOS2: 01 CLKOS3: 10
Provide Standby Port	Adds a STDBY port to power down the PLL.
Provide PLL Reset	Adds a port to reset the PLL.
Provide PLL Lock Signal	Adds an active-high output signal that indicates PLL lock. Select **PLL Lock is Sticky** to make the lock signal sticky.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysCLOCK PLLs and DLLs" on page 1168
- TN1263, *ECP5 sysCLOCK PLL/DLL Design and Usage Guide*

PLL Options (LatticeECP/EC, LatticeXP, MachXO)

Output frequencies depend on the input frequency and the divider values. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

Table 256: PLL Dialog Box for LatticeEC, LatticeECP, LatticeXP, and MachXO

Feature	Description
Frequency Mode	Default mode. When input and clock frequencies are entered in this mode, divider settings are automatically calculated.
Divider Mode	Advanced mode. This mode is for advanced users. Enter input clock frequency and divider values to achieve expected output frequency.
CLKI	CLKI is the input reference clock for the PLL. Specify its frequency in MHz. If in Divider mode, also choose a divider value to achieve the desired output frequency.
CLKOP	**Divider** – In the Frequency mode, divider values cannot be selected. Clicking the "Calculate" button displays the actual values used by the generated module. In the Divider mode, the user can select the CLKOP divider from the drop-down. The listed divider values are not limited when the other clock outputs are enabled. **Desired Frequency** – In the Frequency mode, specify the output clock (CLKOP) frequency. **Tolerance** – Specify a tolerance for the CLKOP frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the "Calculate" button displays the actual frequency used by the generated module.
CLKFB	**Feedback Mode** – Specify the feedback mode to be internal feedback, CLKOP/ CLKOS feedback (to compensate the output clock), or an external user clock. **Divider** – In the divider mode, the divider in the feedback path can be selected from the drop-down list. In the Frequency mode, clicking the "Calculate" button displays the divider value.
CLKOS	**Enable CLKOS** – Selecting this option will produce the CLKOS port in the generated module. This clock has the same frequency as CLKOP and it has the specified phase relation with CLKOP. **PhaseADJ (degree)** – Select the phase difference between CLKOP and CLKOS. **Duty Cycle (*1/8)** – Specify a duty cycle of CLKOS in terms of 1/8th of the period. The legal values are 1 – 7.
Delay Adjust	**Dynamic Delay Control** – Specify if Dynamic delay control is required for the PLL. Selecting this option will generate input ports to control the delay.

Table 256: PLL Dialog Box for LatticeEC, LatticeECP, LatticeXP, and MachXO (Continued)

Feature	Description
CLKOK	**Enable CLKOK** – Selecting this option will produce the CLKOK port in the generated module. This clock is a slower clock than CLKOP. In the Frequency mode, select the desired frequency. In the Divider mode, select the divider value for the desired frequency. **Divider** – In the Frequency mode, divider values cannot be selected. Clicking the "Calculate" button displays the actual values used by the generated module. In the Divider mode, the user can select the CLKOK divider from the drop-down. The listed divider values are not limited when the other clock outputs are enabled. **Desired Frequency** – In the Frequency mode, specify the CLKOK frequency. This is disabled in the divider mode. **Tolerance** – Specify a tolerance for the CLKOK frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the "Calculate" button displays the actual frequency that the PLL can produce.
Calculate	**Frequency Mode**: Software calculates Divider settings based on input/output frequency. **Divider Mode**: Software calculates output frequencies based on divider values.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysCLOCK PLLs and DLLs" on page 1168
- TN1049, *LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide*
- TN1089, *MachXO sysCLOCK Design and Usage Guide*

PLL Options (LatticeECP2/M)

Output frequencies depend on the input frequency and the divider values. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

Table 257: PLL Dialog Box for LatticeECP2 and LatticeECP2M

Feature	Description
Frequency Mode	**Default mode** – When input and clock frequencies are entered in this mode, divider settings are automatically calculated.
Divider Mode	**Advanced mode** – This mode is for advanced users. Enter input clock frequency and divider values to achieve expected output frequency.
CLKI	CLKI is the input reference clock for the PLL. Specify its frequency in MHz. If in Divider mode, also choose a divider value to achieve the desired output frequency.
CLKFB	**Feedback Mode** – Specify the feedback mode to be internal feedback, CLKOP feedback (to compensate the output clock), or an external user clock. **Divider** – In the divider mode, select the divider in the feedback path from the drop-down list. In the frequency mode, the divider value is shown when the “Calculate” button is clicked.
PLL Phase & Duty Options	Allows you to select **Static Mode** or **Dynamic Mode**.
Delay Adjust	Specify if you need dynamic, static, or no delay. If no delay, also specify whether the PLL should go on a GPLL or SPLL site. This choice, in combination with the CLKI setting (above), may require an external capacitor (see External Capacitor Pin following). Check the PAR report for the PLL placement and external capacitor requirement information. ▶ GPLL (Dynamic Delay): Select this option if you require dynamic fine delay adjustment. This configuration will be placed on a GPLL site using an EHXPLLD primitive. ▶ GPLL (Static Delay): Select this option if you require static fine delay adjustment. This configuration will be placed on a GPLL site using an EHXPLLD primitive. ▶ GPLL (No Delay): Select this option if you do not require fine delay adjustment and want the PLL to be placed on a GPLL. This configuration will be placed on a GPLL site using an EHXPLLD primitive. ▶ SPLL (No Delay): Select this option if you do not require fine delay adjustment and want the PLL to be placed on an SPLL. This configuration will be placed on an SPLL site using an EPLLD primitive. ▶ SPLL or GPLL (No Delay): Select this option if you do not require fine delay adjustment. This configuration can be placed on either an SPLL or a GPLL site using an EPLLD primitive.
External Capacitor Pin	The need for an external capacitor is decided automatically depending on the Delay Adjust and CLKI settings (above).

Table 257: PLL Dialog Box for LatticeECP2 and LatticeECP2M (Continued)

Feature	Description
CLKOP	**Bypass (CLKOP=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. Bypassing CLKOP disables its frequency and tolerance and removes it as a feedback mode option. If both CLKOP and CLKOS are in BYPASS, everything is disabled except the CLKI frequency option. **Divider** – In the Frequency mode, divider values cannot be selected. Clicking the "Calculate" button displays the actual values used by the generated module. In the Divider mode, the user can select the CLKOP divider from the drop-down. The listed divider values are not limited when the other clock outputs are enabled. **Desired Frequency** – In the Frequency mode, specify the output clock (CLKOP) frequency. **Tolerance** – Specify a tolerance for the CLKOP frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the "Calculate" button displays the actual frequency that the PLL can produce.
CLKOS	**Enable CLKOS** – Selecting this option will produce the CLKOS port in the generated module and allow you to select Phase and Duty Options. This clock has the same frequency as CLKOP and it has the specified phase relation with CLKOP. **Bypass (CLKOS=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. Bypassing CLKOS disables its frequency, tolerance, and phase shift and removes it as a feedback option. CLKOS's frequency automatically matches CLKOP's frequency. If both CLKOP and CLKOS are in BYPASS, then everything is disabled except the CLKI frequency option. **Phase & Duty Options (Static Mode)** ▶ **Phase Shift (degree)** – Allows phase shift selection in 22.5-degree increments from 0 to 360. ▶ **Duty Cycle (*1/16)** – Allows duty cycle selection in 1/16 increments. **Phase & Duty Options (Dynamic Mode)** See TN1103, "LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide." ▶ **Dynamic Phase with 50% Duty** – Implements dynamic phase shift with 50% duty option. ▶ **Dynamic Phase with Dynamic Duty** – Implements dynamic phase shift with dynamic duty option.

Table 257: PLL Dialog Box for LatticeECP2 and LatticeECP2M (Continued)

Feature	Description
CLKOK	**Enable CLKOK** – Selecting this option will produce the CLKOK port in the generated module. This clock is a slower clock than CLKOP. In the Frequency mode, select the desired frequency. In the Divider mode, select the divider value for the desired frequency. **Bypass (CLKOK=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. **Divider** – In the Frequency mode, divider values cannot be selected. Clicking the "Calculate" button displays the actual values used by the generated module. In the Divider mode, the user can select the CLKOK divider from the drop-down. The listed divider values are not limited when the other clock outputs are enabled. **Desired Frequency** – In the Frequency mode, specify the CLKOK frequency. This is disabled in the Divider mode. **Tolerance** – Specify a tolerance for the CLKOK frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the "Calculate" button displays the actual frequency that the PLL can produce. **Provide CLKOK Divider Reset** – Specify if CLKOK Divider Reset is required as an input port. Selecting this option and enabling CLKOK generates an RSTK input port.
Provide PLL Reset	Provides a reset pin to reset the PLL.
Calculate	**Frequency Mode** – Software calculates Divider settings based on input/output frequency. **Divider Mode** – Software calculates output frequencies based on divider values.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysCLOCK PLLs and DLLs" on page 1168
- TN1103, *LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide*

PLL Options (LatticeECP3)

Output frequencies depend on the input frequency and the divider values. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

Table 258: PLL Dialog Box for LatticeECP3

Feature	Description
Frequency Mode	**Default mode** – In this mode, entering input and clock frequencies will automatically calculate divider settings.
Divider Mode	**Advanced mode** – This mode is for advanced users. Enter input clock frequency and divider values to achieve expected output frequency.
CLKI	CLKI is the input reference clock for the PLL. Specify its frequency in MHz. If in Divider mode, also choose a divider value to achieve the desired output frequency. After clicking Calculate, the approximate bandwidth is calculated and displayed here.
CLKFB	**Feedback Mode** – Specify the feedback mode to be internal feedback, CLKOP feedback (to compensate the output clock), CLKOS, or an external user clock. **Divider** – In Divider mode, the divider in the feedback path can be selected from the drop-down list. In Frequency mode, the divider value is shown when the “Calculate” button is clicked.
PLL Phase & Duty Options	Allows you to select **Static Mode** or **Dynamic Mode**.
Provide PLL Reset	Provides a reset pin to reset the PLL.
Provide FINDELA Port	Specify a dynamic single-step delay adjustment signal for CLKOS.
Enable CLKOK2(CLKOP/3)	Specify a clock that is equal to CLKOP/3.
CLKOP/CLKOS (Non Bypass Mode)	These features are not available when both CLKOP and CLKOS are in Bypass mode (below). **Divider** – In Frequency mode, divider values cannot be selected. Clicking the “Calculate” button displays the actual values used by the generated module. In Divider mode, select the CLKOP divider from the drop-down. The listed divider values are not limited when the other clock outputs are enabled. **Desired Frequency** – In Frequency mode, specify the output clock (CLKOP) frequency. **Tolerance** – Specify a tolerance for the CLKOP frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the “Calculate” button displays the actual frequency that the PLL can produce.

Table 258: PLL Dialog Box for LatticeECP3 (Continued)

Feature	Description
CLKOP	**Bypass (CLKOP=CLKI)** – The bypass option connects the output to the input, bypassing the PLL circuit. Bypassing CLKOP and CLKOS disables the CLKOP desired frequency and tolerance fields and removes CLKOP as a feedback mode option. If both CLKOP and CLKOS are in BYPASS, then everything is disabled except the CLKI frequency option. **Duty Trim Option (Dynamic Mode)** ▶ **Rising** – This selection applies to the rising edge. ▶ **Falling** – This selection applies to the falling edge. ▶ **Delay Multiplier** – This selection provides delay multiplier for the rising edge or falling edge duty trim option.
CLKOS	**Enable CLKOS** – Selecting this option will produce the CLKOS port in the generated module. This clock has the same frequency as CLKOP and it has the specified phase relation with CLKOP. **Bypass (CLKOS=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. Bypassing CLKOS disables its frequency, tolerance, and phase shift fields and removes it as a feedback option. CLKOS's frequency automatically matches CLKOP's frequency. **Phase & Duty Options (Static Mode)** ▶ **Phase Shift (degree)** – Allows phase shift selection in 22.5-degree increments from 0 to 360. ▶ **Duty Cycle (*1/16)** – Allows duty cycle selection in 1/16 increments. ▶ **Delay Multiplier** – This selection is the delay multiplier for the rising edge or falling edge. **Phase & Duty Options (Dynamic Mode)** See TN1103, "LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide." ▶ **Dynamic Phase with 50% Duty** – Implements dynamic phase shift with 50% duty option. ▶ **Dynamic Phase with Dynamic Duty** – Implements dynamic phase shift with dynamic duty option. **Duty Trim Options (Dynamic Mode)** ▶ **Rising** – This selection applies to the rising edge. ▶ **Falling** – This selection applies to the falling edge. ▶ **Delay Multiplier** – This selection is the delay multiplier for the rising edge or falling edge duty trim option.

Table 258: PLL Dialog Box for LatticeECP3 (Continued)

Feature	Description
CLKOK	**Enable CLKOK** – Selecting this option produces the CLKOK port in the generated module. This clock is a slower clock than CLKOP. In frequency mode, select the desired frequency. In divider mode, select the divider value for the desired frequency. **Bypass (CLKOK=CLKI)** – The bypass option connects the output to the input, bypassing the PLL circuit. **CLKOP as Input/CLKOS as Input** – If CLKOS is enabled (above), you have a choice between using CLKOP or CLKOS as the input for CLKOK. If CLKOS is not enabled, CLKOP is used as the input. **Divider** – In Frequency mode, this divider value cannot be selected. Clicking the "Calculate" button displays the actual values used by the generated module. In Divider mode, select the CLKOK divider from the drop-down list. The listed divider values are not limited when the other clock outputs are enabled. **Desired Frequency** – In Frequency mode, specify the CLKOK Frequency. This is disabled in Divider mode. **Tolerance** – Specify a tolerance for the CLKOK frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the "Calculate" button displays the actual frequency that the PLL can produce. **Provide CLKOK Divider Reset** – Specify if CLKOK Divider Reset is required as an input port. When this is selected and CLKOK is enabled, an input port RSTK is generated.
Calculate	**Frequency Mode**: Software calculates Divider settings based on input/output frequency. **Divider Mode**: Software calculates output frequencies based on divider values.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysCLOCK PLLs and DLLs" on page 1168
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide

PLL Options (LatticeSC/M)

Output frequencies depend on the input frequency and the divider values. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

Table 259: PLL Dialog Box for LatticeSC/M

Feature	Description
CLKI	Specify the frequency of the input clock in MHz. This should be in the range of 2 - 1000 MHz.
CLKFB	**Feedback Mode** – Specify the feedback mode to be internal feedback, CLKOP/ CLKOS feedback (to compensate the output clock), or an external user clock. **Frequency** – If the CLKFB Feedback Mode is set to User Clock, specify the frequency in MHz.
Provide RSTN Port	Specify if a RSTN port is required in the generated PLL. A reset port can shut off the output clock and have undesirable effects on the system.
Provide SMI Ports on PLL Module	Specify if the SMI port that allows runtime reconfiguration of the PLL block using the SMI ports is required in the generated PLL.
Enable High Bandwidth	Use the Enable High Bandwidth option to increase the bandwidth of the PLL and reduce the noise floor of the PLL. The trade-off is that this will cause more jitter peaking of the PLL.
CLKOP	**Bypass (CLKOP=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. Bypassing CLKOP disables its frequency and tolerance fields and removes it as a feedback mode option. **Desired Frequency** – In the frequency mode, specify the output clock (CLKOP) Frequency. **Tolerance** – Specify a tolerance for the CLKOP frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Specify the desired frequency of the output clock, CLKOP. This can be in the range of 1.5625 MHz to 1 GHz. The tolerance is specified as a percentage of the desired frequency. The actual frequency that the PLL can produce is shown when the “Calculate” button is clicked.

Table 259: PLL Dialog Box for LatticeSC/M (Continued)

Feature	Description
CLKOS	**Bypass (CLKOS=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. Bypassing CLKOS disables its frequency, tolerance, and phase shift fields, and removes it as a feedback option. If both CLKOP and CLKOS are in BYPASS, then everything is disabled except the CLKI frequency option. **Desired Frequency** – In the Frequency mode, specify the output clock (CLKOP) Frequency. **Tolerance** – Specify a tolerance for the CLKOP frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of value, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Specify the desired frequency of the output clock, CLKOP. This can be in the range of 1.5625 MHz to 1 GHz. The tolerance is specified as a percentage of the desired frequency. Clicking the “Calculate” button displays the actual frequency that the PLL can produce. **Phase shift (degree)** – Specify the desired phase shift of the CLKOS output compared to the CLKOP output. The CLKOS output will be delayed this amount from the CLKOP output. Two attributes determine the total phase shift (PHASEADJ and CLKOS_VCODEL). These attributes are based on the VCO rate of the PLL. IPexpress will perform the calculations and set these attributes to correctly produce the specified phase shift on the CLKOS output.
Info	**VCO Rate (MHz)** – The VCO rate in MHz is automatically updated by clicking the Calculate button. **Approximate PLL Bandwidth (MHz)** – The approximate PLL bandwidth in MHz is automatically updated by clicking the Calculate button.
Calculate	The Calculate button updates the actual frequency the PLL will generate. Not all values can be generated, due to a limited set of divider values and other hardware limitations. This also updates the VCO Rate and Approximate PLL Bandwidth boxes.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with sysCLOCK PLLs and DLLs” on page 1168
- TN1098 - LatticeSC sysCLOCK PLL/DLL User's Guide

PLL Options (LatticeXP2)

Output frequencies depend on the input frequency and the divider values. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

Table 260: PLL Dialog Box for LatticeXP2

Feature	Description
Frequency Mode	**Default mode** – In this mode, entering input and clock frequencies automatically calculates the divider settings.
Divider Mode	**Advanced mode** – This mode is for advanced users. Enter input clock frequency and divider values to achieve expected output frequency.
CLKI	CLKI is the input reference clock for the PLL. Specify its frequency in MHz. If in Divider mode, also choose a divider value to achieve the desired output frequency.
CLKFB	**Feedback Mode** – Specify the feedback mode to be internal feedback, CLKOP feedback (to compensate the output clock), or an external user clock. **Divider** – In the Divider mode, you can select the divider in the feedback path from the drop-down list. In the Frequency mode, clicking the “Calculate” button displays the divider values.
PLL Phase & Duty Options	Allows you to select **Static Mode** or **Dynamic Mode**.
Enable CLKOK2(CLKOP/3)	Enables the CLKOK2 PLL output. The frequency of CLKOK2 is one-third the frequency of CLKOP. The CLKOK2 is in phase with CLKOP.
Provide PLL Reset	Provides a reset pin to reset the PLL.
Provide Dynamic Power Down Port	Adds a signal that starts the powerdown routine.
Calculate	**Frequency Mode**: Software calculates Divider settings based on input/output frequency. **Divider Mode**: Software calculates output frequencies based on divider values.
CLKOP	**Bypass (CLKOP=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. **Divider** – You cannot select divider values in Frequency mode. Clicking the “Calculate” button displays the actual values used by the generated module. In Divider mode, you can select the CLKOP divider from the drop-down menu. **Desired Frequency** – In Frequency mode, specify the output clock frequency. **Tolerance** – Specify a tolerance for the frequency, as a percentage of the requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the “Calculate” button displays the actual frequency that the PLL will produce. **Duty Trim Option (Dynamic Mode)** – Choose a delay multiplier for the rising edge or falling edge duty trim option.

Table 260: PLL Dialog Box for LatticeXP2

Feature	Description
CLKOS	**Enable CLKOS** – Selecting this option will produce the CLKOS port in the generated module. This clock has the same frequency as CLKOP and it has the specified phase relation with CLKOP. **Bypass (CLKOS=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. **Phase & Duty Options (Static Mode)** ▶ **Phase Shift (degree)** – Allows phase shift selection in 22.5-degree increments from 0 to 360. ▶ **Duty Cycle (*1/16)** – Allows duty cycle selection in 1/16 increments. **Phase & Duty Options (Dynamic Mode)** ▶ **Dynamic Phase Shift with 50% Duty** – Implements dynamic phase shift with 50% duty option. See tech note, “LatticeXP2 sysCLOCK PLL/DLL Design and Usage Guide” for details. ▶ **Dynamic Phase with Dynamic Duty** – Implements dynamic phase shift with dynamic duty option. See tech note, “LatticeXP2 sysCLOCK PLL/DLL Design and Usage Guide” for details. **Duty Trim Options (Dynamic Mode)** ▶ **Rising** – This selection applies to the rising edge. ▶ **Falling** – This selection applies to the falling edge. ▶ **Delay Multiplier** – This selection provides delay multiplier for the rising edge or falling edge duty trim option. **Provide CLKOS Fine Delay** – Adds a signal to add approximately 70 ps of delay.
CLKOK	**Enable CLKOK** – Selecting this option will produce the CLKOK port in the generated module. This clock is a slower clock than CLKOP. In the Frequency mode, select the desired frequency. In the Divider mode, select the divider value for the desired frequency. **Bypass (CLKOK=CLKI)** – The bypass option will connect the output to the input, bypassing the PLL circuit. **Divider** – In the Frequency mode, divider values cannot be selected. Clicking the “Calculate” button displays the actual values used by the generated module. In the Divider mode, the user can select the CLKOK divider from the drop-down. The listed divider values are not limited when the other clock outputs are enabled. **Desired Frequency** – In the Frequency mode, specify the CLKOK Frequency. This is disabled in the Divider mode. **Tolerance** – Specify a tolerance for the CLKOK frequency, as a percentage of requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – Clicking the “Calculate” button displays the actual frequency that the PLL can produce. **Provide CLKOK Divider Reset** – Specify if CLKOK Divider Reset is required as an input port. Selecting this option and enabling CLKOK generates an RSTK input port.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.

Table 260: PLL Dialog Box for LatticeXP2

Feature	Description
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysCLOCK PLLs and DLLs" on page 1168
- TN1126, *LatticeXP2 sysCLOCK PLL Design and Usage Guide*

PLL Options (MachXO2, MachXO3L, Platform Manager 2)

Available with MachXO2 with 1200 LUTs or larger, MachXO2-640UHC, MachXO3L, and Platform Manager 2-21.

Output frequencies depend on the input frequency and the divider values. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

The Calculate button is just below the CLKOS3 box. If you do not see the Calculate button, scroll down.

Table 261: PLL Dialog Box for MachXO2, MachXO3L, and Platform Manager 2

Feature	Description
Frequency Mode	**Default mode** – In this mode, entering input and clock frequencies automatically calculates Divider settings.
Divider Mode	**Advanced mode** – This mode is for advanced users. Enter input clock frequency and divider values to achieve expected output frequency.
CLKI	CLKI is the input reference clock for the PLL. Specify its frequency in MHz. If in Divider mode, also choose a divider value to achieve the desired output frequency. After clicking Calculate, the approximate bandwidth and VCO frequency are calculated and displayed here.
CLKFB	**FBK Mode** – Specify the feedback clock source. Choices are internal and external CLKOP, CLKOS, CLKOS2, CLKOS3, and external user clock. To use CLKOS, CLKOS2, or CLKOS3, select the clock's Enable option and then choose the FBK mode. If you select Bypass for CLKOP, the CLKOP, INT_OP, and UserClock options are disabled. **FBK Divider** – In the Divider mode, the divider in the feedback path can be selected from the drop-down list. In the Frequency mode, clicking the "Calculate" button displays the divider value. **Fractional-N Divider** – To finely divide the feedback clock frequency, select **Enable** and enter a divider value. Available in both Frequency and Divider modes.

Table 261: PLL Dialog Box for MachXO2, MachXO3L, and Platform Manager 2 (Continued)

Feature	Description
Optional Port Selections	Select additional ports for the module. **Dynamic Phase Ports** – Adds phase controls. The PHASESEL port selects a clock with the following code: CLKOP: 11 CLKOS: 00 CLKOS2: 01 CLKOS3: 10 **Clock Enable Ports** – Adds an enable input port for each selected clock. **Standby Ports** – Adds a STDBY port to power down the PLL. **Enable Clock Select** – Adds a second clock and a signal to select which one to use.
PLL Reset Options	Select reset options for the PLL. Each of the following adds an active-high input port that resets a different clock: **Provide PLL Reset** – Resets CLKOP. **Provide PLLM Reset** – Resets CLKOP and the CLKI divider. **PLL CLKOS2 Reset** – Resets the CLKOS2 divider. **PLL CLKOS3 Reset** – Resets the CLKOS3 divider.
Lock Settings	Adds an active-high output signal that indicates PLL lock. Select **PLL Lock is Sticky** to make the lock signal sticky.
Wishbone BUS	Adds ports allowing you to configure the PLL through a Wishbone bus: PLLADDR[4:0]: Address (input) PLLCLK: Clock (input) PLLDATI[7:0]: Data (input) PLLRST: Reset for data bus, not any registers (input) PLLSTB: Strobe (input) PLLWE: write enable (input) PLLACK: Acknowledge (output) PLLDATO[7:0]: Data (output) Use with an EFB module with the PLL (Dynamic access) option. See “EFB” on page 1448.

Table 261: PLL Dialog Box for MachXO2, MachXO3L, and Platform Manager 2 (Continued)

Feature	Description
CLKOP, CLKOS, CLKOS2, CLKOS3	The PLL can produce up to four different clock signals. Select **Enable** for the clocks you want to use (CLKOP is always enabled). Then use the following options to specify the form of the clocks. The duty trim option is only available with CLKOP and CLKOS. **Enable** – Select to add the clock to the module. CLKOP is the standard output and is always enabled. **Bypass** – The bypass option connects the output to the input, bypassing the PLL circuit. However, you can still apply a divider to the frequency. See "Clock Divider" below. **Clock Divider** – If you are applying the bypass option to the clock, you can still apply a divider to the frequency. But you must be using Divider mode for the PLL. To apply a divider, select **Clock Divider** and then choose a value from the Divider drop-down menu. **Desired Frequency** – In Frequency mode, specify the output clock frequency. **Tolerance %** – Specify a tolerance for the frequency as a percentage of the requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Divider** – Select a value to divide the input frequency by. Only available in Divider mode. To see the actual output frequency, click "Calculate" and check the Actual Frequency box. **Actual Frequency** – Clicking "Calculate" displays the actual frequency that the PLL will produce. **Static Phase Shift (degree)** – Choose a phase shift to build into the PLL. **Duty Trim Options** – (CLKOP and CLKOS only) Choose a delay multiplier for the rising edge or falling edge duty trim option.
Calculate	**Frequency Mode** – Software calculates divider settings based on input/output frequency. **Divider Mode** – Software calculates output frequencies based on divider values.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysCLOCK PLLs and DLLs" on page 1168
- TN1199, *MachXO2 sysCLOCK PLL Design and Usage Guide*
- TN1282, *MachXO3L sysCLOCK PLL Design and Usage Guide*

PLL Options (LIFMD)

Also applicable to LIFMDF.

The module dialog box has three tabs:

- Frequency, see below
- Phase, see Table 254 on page 1576
- Optional Ports, see Table 255 on page 1576

Output frequencies depend on the input frequency, the desired output frequencies, and the tolerances. To see what the actual output frequencies will be, set the options and then click **Calculate**. If the results are not satisfactory, adjust the values and recalculate before generating the module.

After setting up the Frequency tab, go to the other tabs to specify phase shifts for the output clocks and optional ports such as for dynamic phase shift, clock enable, and PLL reset.

Table 262: PLL Dialog Box for LIFMD/F—Frequency Tab

Feature	Description
CLKI	**Frequency** – CLKI is the input reference clock for the PLL. Specify its frequency in megahertz. **Refclk Divider** – After clicking Calculate, this shows the divider value used to achieve the desired output frequency. **Enable High Bandwidth** – Select to have a higher PLL bandwidth. After clicking Calculate, the bandwidth is shown in the Info box. **PLL Reference Clock from Pin** – Select to get the reference clock directly from one of the FPGA's pins. Then choose the pin's buffer type.
CLKFB	**Feedback Mode** – Specify the feedback clock source. Choices are internal and external CLKOP, CLKOS, CLKOS2, CLKOS3, INT_OP, INT_OS, INT_OS2, INT_OS3, UserClock. To use CLKOS, CLKOS2, or CLKOS3, select the clock's **Enable** option and clear the **Bypass** option. Then choose the feedback mode. If you select Bypass for CLKOP, the CLKOP, INT_OP, and UserClock options are disabled. **Feedback Divider** – The divider in the feedback path is shown after clicking Calculate.
Info	These values are shown after filling in the other fields and clicking Calculate.

Table 262: PLL Dialog Box for LIFMD/F—Frequency Tab (Continued)

Feature	Description
CLKOP, CLKOS, CLKOS2, CLKOS3	The PLL can produce up to four different clock signals. Select **Enable** for the clocks you want to use (CLKOP is always enabled). Then use the following options to specify the form of the clocks. **Enable** – Select to add the clock to the module. CLKOP is the standard output and is always enabled. **Bypass** – The bypass option connects the output to the input, bypassing the PLL circuit. **Output Divider** – Shows the calculated divider to produce the actual frequency. **Desired Frequency** – Specify the desired output clock frequency. **Tolerance %** – Specify a tolerance for the frequency as a percentage of the requested frequency. Since the dividers can only take a certain set of values, not all frequencies can be generated. The tolerance value is used to guide the tool to select a suitable divider value. **Actual Frequency** – The actual frequency that the PLL will produce is after clicking Calculate.
Calculate	Software calculates divider settings and actual output frequencies based on CLKI and desired frequencies.
Configure	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 263: PLL Dialog Box for LIFMD/F—Phase Tab

Feature	Description
Desired Phase	Specify the desired phase shift in degrees.This option is available if the clock is enabled but the Bypass option has not been selected and the clock has not been selected in Feedback Mode.
Actual Phase	Shows the actual phase shift after clicking Calculate. Not all phase shifts are possible.
Calculate	After entering the desired phase shifts, click **Calculate** to see what actual phase shift will be implemented.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 264: PLL Dialog Box for LIFMD/F—Optional Ports Tab

Feature	Description
Enable Clock Select	Adds a second input clock port, CLKI2, and a select port.
Dynamic Phase Ports	Adds phase controls.
Clock Enable OP, OS, OS2, OS3	Adds an enable input port for each selected clock. This option is available if the clock is enabled but the Bypass option has not been selected and the clock has not been selected in Feedback Mode. The PHASESEL port selects a clock with the following code: CLKOP: 11 CLKOS: 00 CLKOS2: 01 CLKOS3: 10
Provide Standby Port	Adds a STDBY port to power down the PLL.
Provide PLL Reset	Adds a port to reset the PLL.
Provide PLL Lock Signal	Adds an active-high output signal that indicates PLL lock. Select **PLL Lock is Sticky** to make the lock signal sticky.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysCLOCK PLLs and DLLs" on page 1168
- TN1304, *CrossLink sysCLOCK PLL/DLL Design and Usage Guide*

Spmi_pll

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_pll.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)	Notes
I	CLKI	Bit	N/A	
I	CLKFB	Bit	N/A	
I	RESET	Bit	N/A	Inverted sense in LatticeSC/M.

Input/Output	Port Name	Type	Size (Buses Only)	Notes
O	LOCK	Bit	N/A	
O	CLKOP	Bit	N/A	
O	CLKOS	Bit	N/A	
O	CLKOK	Bit	N/A	Not applicable for LatticeSC/M.
O	CLKOK2	Bit	N/A	LatticeECP3 and LatticeXP only.

Verilog pmi_pll Definition

```
module pmi_pll
 #(parameter pmi_freq_clki = 100,
   parameter pmi_freq_clkfb = 100,
   parameter pmi_freq_clkop = 100,
   parameter pmi_freq_clkos = 100,
   parameter pmi_freq_clkok = 100,
   parameter pmi_family = "SC",
   parameter pmi_phase_adj = 0,
   parameter pmi_duty_cycle = 50,
   parameter pmi_clkfb_source = "CLKOP",
   parameter pmi_fdel = "off",     // ECP2 Only
   parameter pmi_fdel_val = 0,
   parameter module_type = "pmi_pll")

  (input       CLKI,
   input       CLKFB,
   input       RESET,
   output      CLKOP,
   output      CLKOS,
   output      CLKOK,
   output      CLKOK2,
   output      LOCK)/*synthesis syn_black_box */;

endmodule //pmi_pll
```

VHDL pmi_pll Definition

```
component pmi_pll is
     generic (
       pmi_freq_clki : integer := 100;
       pmi_freq_clkfb : integer := 100;
       pmi_freq_clkop : integer := 100;
       pmi_freq_clkos : integer := 100;
       pmi_freq_clkok : integer := 100;
       pmi_family : string := "SC";
       pmi_phase_adj : integer := 0;
       pmi_duty_cycle : integer := 50;
       pmi_clkfb_source : string := "CLKOP";
       pmi_fdel : string := "off";
       pmi_fdel_val : integer := 0;
       module_type : string := "pmi_pll"
    );
    port (
     CLKI: in std_logic;
     CLKFB: in std_logic;
     RESET: in std_logic;
```

```
    CLKOP: out std_logic;
    CLKOS: out std_logic;
    CLKOK: out std_logic;
    CLKOK2: out std_logic;
    LOCK: out std_logic
  );
end component pmi_pll;
```

Parameter Names and Values

The following table contains the pmi_pll module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_freq_clki	15 to 1000 MHz (LatticeSC/M) 1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	100	
pmi_freq_clkfb	15 to 1000 MHz (LatticeSC/M) 1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	100	
pmi_freq_clkop	1.562 to 1000 MHz (LatticeSC/M) 1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	100	
pmi_freq_clkok	1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	50	Not available for LatticeSC/M.
pmi_freq_clkos	1 to 840 MHz (LatticeSC/M)	100	LatticeSC/M only.
pmi_phase_adj	0, 22.5, 45, 90, …, 315, 337.5 (LatticeECP2, LatticeECP3, LatticeXP2) 0, 45, 90, …, 315 (LatticeECP/EC, LatticeSC/M, MachXO)	0	
pmi_duty_cycle	12.5, 25, 37.5, 62.5, 75, 87.5 (LatticeECP/EC, MachXO) 12.5, 18.75, 25, 31.25, 37.5, 43.75, 50, 56.25, 62.5, 68.75, 75, 81.25, 87.5 (LatticeECP2, LatticeECP3, LatticeXP)	50	▶ Listed as a percentage. ▶ Does not apply to LatticeSC/M.
pmi_clkfb_source	INTERNAL, CLKOP, USERCLOCK, CLKOS	CLKOP	The CLKOS option is only applicable for LatticeECP3.
pmi_fdel	“on” | “off”	“off”	LatticeECP2 only.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_fdel_val	-8 to 8	0	LatticeECP/EC, LatticeECP2/M, and LatticeXP only.
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L | “LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

pmi_pll_fp

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_pll_fp (PLL Floating Point).

Port Description

Input/Output	Port Name	Type	Size (Buses Only)	Notes
I	CLKI	Bit	N/A	
I	CLKFB	Bit	N/A	
I	RESET	Bit	N/A	Inverted sense in LatticeSC/M.
O	LOCK	Bit	N/A	
O	CLKOP	Bit	N/A	
O	CLKOS	Bit	N/A	
O	CLKOK	Bit	N/A	Not applicable for LatticeSC/M.
O	CLKOK2	Bit	N/A	LatticeXP2, and LatticeECP3 only.

Verilog pmi_pll_fp Definition

```
module pmi_pll_fp
 #(parameter pmi_freq_clki = "100.0",
   parameter pmi_freq_clkfb = "100.0",
   parameter pmi_freq_clkop = "100.0",
   parameter pmi_freq_clkos = "100.0",
   parameter pmi_freq_clkok = "50.0",
   parameter pmi_family = "EC",
   parameter pmi_phase_adj = "0.0",
   parameter pmi_duty_cycle = "50.0",
   parameter pmi_clkfb_source = "CLKOP",
   parameter pmi_fdel = "off",      // ECP2 Only
   parameter pmi_fdel_val = 0,
   parameter module_type = "pmi_pll_fp")

  (input       CLKI,
```

```
   input       CLKFB,
   input       RESET,
   output      CLKOP,
   output      CLKOS,
   output      CLKOK,
   output      CLKOK2,
   output      LOCK)/*synthesis syn_black_box */;
endmodule //pmi_pll_fp
```

VHDL pmi_pll_fp Definition

```
component pmi_pll_fp is
     generic (
       pmi_freq_clki : string := "100.0";
       pmi_freq_clkfb : string := "100.0";
       pmi_freq_clkop : string := "100.0";
       pmi_freq_clkos : string := "100.0";
       pmi_freq_clkok : string := "50.0";
       pmi_family : string := "EC";
       pmi_phase_adj : string := "0.0";
       pmi_duty_cycle : string := "50.0";
       pmi_clkfb_source : string := "CLKOP";
       pmi_fdel : string := "off";
       pmi_fdel_val : integer := 0;
       module_type : string := "pmi_pll_fp"
    );
    port (
     CLKI: in std_logic;
     CLKFB: in std_logic;
     RESET: in std_logic;
     CLKOP: out std_logic;
     CLKOS: out std_logic;
     CLKOK: out std_logic;
     CLKOK2: out std_logic;
     LOCK: out std_logic
   );
  end component pmi_pll_fp;
```

Parameter Names and Values

The following table contains the pmi_pll module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_freq_clki	15 to 1000 MHz (LatticeSC/M) 1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	100	
pmi_freq_clkfb	15 to 1000 MHz (LatticeSC/M) 1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	100	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_freq_clkop	1.562 to 1000 MHz (LatticeSC/M) 1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	100	
pmi_freq_clkok	1 to 420 MHz (LatticeECP2, LatticeXP2) 25 to 420 MHz (LatticeECP/EC, LatticeXP) 5 to 500 MHz (LatticeECP3)	50	Not available for LatticeSC/M.
pmi_freq_clkos	1 to 840 MHz (LatticeSC/M)	100	LatticeSC/M only.
pmi_phase_adj	0, 22.5, 45, 90, …, 315, 337.5 (LatticeECP2, LatticeECP3, LatticeXP2) 0, 45, 90, …, 315 (LatticeECP/EC, LatticeSC/M, MachXO)	0	
pmi_duty_cycle	12.5, 25, 37.5, 62.5, 75, 87.5 (LatticeECP/EC, MachXO) 12.5, 18.75, 25, 31.25, 37.5, 43.75, 50, 56.25, 62.5, 68.75, 75, 81.25, 87.5 (LatticeECP2, LatticeECP3, LatticeXP)	50	▶ Listed as a percentage. ▶ Does not apply to LatticeSC/M.
pmi_clkfb_source	INTERNAL, CLKOP, USERCLOCK, CLKOS	CLKOP	The CLKOS option is only applicable for LatticeECP3.
pmi_fdel	“on” | “off”	“off”	LatticeECP2 only
pmi_fdel_val	-8 to 8	0	LatticeECP/EC, LatticeECP2/M, and LatticeXP only.
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” |”ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” | “LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

PMU

The Power Management Unit (PMU) can reduce power consumption for Lattice FPGA products.

Table 265: PMU Dialog Box

Feature	Description
Device Wake Up(Sleep Mode to Normal Mode) Options	**External User Wake Up** – Allows signals from an external pin to wake up the PMU. **User I2C Wake Up** – Allows signals from an I2C0 component to wake up the PMU. **Watch Dog Timer (WDT) Expiry Wake Up** – Allows signals from a Watch Dog Timer to wake up the PMU.
Watch Dog Timer Expiry Wake Up Options	Select which Watch Dog Timer should wake up the PMU.
Watch Dog Timer Initialization Options	Enter the time interval in hours (HH), minutes (MM), and seconds (SS) after which the Watch Dog Timer will wake the PMU.
Watch Dog Timer(WDT) User Mode Options	**Enable User Mode for Watch Dog Timer in Normal State** – Allows you to use Watch Dog Timers in Normal State. **Mode Selection** – Select Count Once to complete a single countdown, or Count Repetitively to start a new timer every time the current one finishes. **Select the Watch Dog Timer (WDT) Counter that will be used in User Mode** – Select which Watch Dog Timer should wake up the PMU in User Mode. **Interrupt Enable** – Adds a Watch Dog Timer Interrupt port to a Normal State design.
Configure	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195

Power Controller

Power Controller ensures smooth transitions into and out of standby mode.

Table 266: Power Controller Dialog Box

Feature	Description
Entry Signals	Select entry options. Options include User, Configuration, and User & Configuration. Configuration includes JTAG, SPI, and I2C.
Stop to Standby Delay	Select how to control the delay between the STOP and STDBY signals: User: Adds the USERTIMEOUT enable signal for Power Controller's timer counter. Counter: Adds the clock signal for Power Controller's timer counter. Bypass: No delay.

Table 266: Power Controller Dialog Box

Feature	Description
Wake Signals	Specify the source of the wake-up signal. Choose User for user logic, or choose Configuration to use an internal JTAG, SPI, or I2C block (see "EFB" on page 1448).
Enable Standby Flags	Includes standby flags.
Turn off POR when in Standby	Turns off the Power-On-Reset circuit when in standby mode. Recommended only for applications with reliable power supply rails.
Turn off Bandgap when in Standby	Turning off Bandgap also turns off POR. Bandgap is required for use of the oscillator, PLL, and referenced I/Os.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- TN1198 - Power Estimation and Management for MachXO2 Devices

Power Guard

Power Guard controls dynamic power consumption by blocking signals at the input buffers. This feature can be used in both clock and data paths. Power Guard's biggest impact is in standby mode when it can be used to switch off clock inputs that are distributed using general routing resources.

Table 267: Power Guard Dialog Box

Feature	Description
Specify the width of the Power Guard bus	Enter the number of bits in the bus to be controlled by the Power Guard module.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.

Table 267: Power Guard Dialog Box

Feature	Description
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- TN1198 - Power Estimation and Management for MachXO2 Devices

RAM_Based_Shift_Register

A shift register using RAM.

Table 268: RAM_Based_Shift_Register Dialog Box

Feature	Description
Data Width	Specify the width of the data.
Ram Type	**LUT Based** – Contains LUTs and logic implementation. **EBR Based** – Contains Embedded Block RAM implementation.
Shift Register Type	Specify the type of shift register. The output data is the input data delayed by a certain number of clock cycles. **Fixed Length** –The delay is fixed and is always equal to the number of shifts. **Number of Shifts** – Specify the number of shifts for a fixed-length shift register. **Variable Length (Lossy)** – The delay is controlled by the Addr input. When the "Addr" changes, the current input data will appear at the output Addr cycles later. The output for the next Addr cycles is not defined. **Variable Length (Lossless)** – The delay is controlled by the Addr input. When the "Addr" changes, the current output data is what was at the input Addr cycles earlier. **Max Number of Shifts** – Specify the maximum number of shifts for a variable length shift register. This will determine the size of the Addr input.
Enable Output Register	Specify that registers are used for the output data. This will produce an additional clock cycle delay for the output.
Memory File	(ECP5, LatticeECP3, LatticeSC/M, LatticeXP2, MachXO2, MachXO3L, and Platform Manager 2 only) Allows you to select a memory initialization (.mem) file for the module. See "Creating a Memory Initialization File with Memory Generator" on page 203.
Memory File Format	(ECP5, LatticeECP3, LatticeSC/M, LatticeXP2, MachXO2, MachXO3L, and Platform Manager 2 only) Allows you to choose Binary, Hex, or Addressed Hex format.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.

Table 268: RAM_Based_Shift_Register Dialog Box (Continued)

Feature	Description
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- TN1264, *ECP5 Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1290, *Memory Usage Guide for MachXO3L Devices*

pmi_distributed_shift_reg

Note

The module name pmi_distributed_shift_reg will change when the configuration is implemented.

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_distributed_shift_reg.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Din	Bus	(pmi_data_width - 1):0
I	Addr	Bus	(pmi_max_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_distributed_shift_reg Definition

```
module pmi_distributed_shift_reg
  #(parameter pmi_data_width = 16,
    parameter pmi_regmode = "reg",
    parameter pmi_shiftreg_type = "fixed",
    parameter pmi_num_shift = 16,
    parameter pmi_num_width = 4,
    parameter pmi_max_shift = 16,
    parameter pmi_max_width = 4,
```

```
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_distributed_shift_reg")

    (
     input [(pmi_data_width-1):0] Din,
     input [(pmi_max_width-1):0] Addr,
     input Clock,
     input ClockEn,
     input Reset,
     output [(pmi_data_width-1):0] Q)/*synthesis
syn_black_box*/;

endmodule // pmi_distributed_shift_reg
```

VHDL pmi_distributed_shift_reg Definition

```
component pmi_distributed_shift_reg is
     generic (
       pmi_data_width : integer := 16;
       pmi_regmode : string := "reg";
       pmi_shiftreg_type : string := "fixed";
       pmi_num_shift : integer := 16;
       pmi_num_width : integer := 4;
       pmi_max_shift : integer := 16;
       pmi_max_width : integer := 4;
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_family : string := "EC";
       module_type : string := "pmi_distributed_shift_reg"
    );
    port (
     Din : in std_logic_vector((pmi_data_width-1) downto 0);
     Addr : in std_logic_vector((pmi_max_width-1) downto 0);
     Clock: in std_logic;
     ClockEn: in std_logic;
     Reset: in std_logic;
     Q : out std_logic_vector((pmi_data_width-1) downto 0)
   );
  end component pmi_distributed_shift_reg;
```

Parameter Names and Values

The following table contains the pmi_distributed_shift_reg module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_data_width	1 to 256	16	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_shiftreg_type	“fixed” | “variable” | “lossless”	“fixed”	
pmi_num_shift	2 to 1024	16	
pmi_num_width	0 to 10	4	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_max_shift	2 to 1024	16	
pmi_max_width	0 to 10	4	
pmi_init_file	<string>	“none”	The pmi_init_file and pmi_initi_file_format parameters are only supported for LatticeSC/M, LatticeECP3, and LatticeXP2 device families in distributed mode.
pmi_init_file_format	“binary” | “hex”	“binary”	The pmi_init_file and pmi_initi_file_format parameters are only supported for LatticeSC/M, LatticeECP3, and LatticeXP2 device families in distributed mode.
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” |“LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

RAM_DP

A pseudo-dual-port RAM that provides a variety of RAM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

The options vary with the device family. See:

- “RAM_DP Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)” on page 1606
- “RAM_DP Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)” on page 1607

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*

- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1306, *CrossLink Memory Usage Guide*

RAM_DP Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)

A pseudo-dual-port RAM that provides a variety of RAM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical note listed in the “See Also” section below.

Table 269: RAM_DP Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2

Feature	Description
Specify the Size of the RAM_DP	Both Read and Write port width can be specified. Enter number of address locations times data width. The two ports can be of different widths. In such cases, the memory initialization file corresponds to the Write Port. For example, for a RAM_DP with Write Port = 1024x36 and Read Port = 2048x18, the initialization file should be of size 1024x36.
Provide Byte Enables	Allows for byte enables at a size of 8 or 9 bytes. Choose the byte size from the drop-down menu.
Enable Output Register	When selected, the output register is enabled (separate controls for Read Port and Write Port).
Enable Output ClockEn	Adds an output clock enable port. This is an input port. Requires that the Output Register (above) be enabled first.
Optimization	Select whether to optimize the module for minimum area or maximum speed.
Reset Mode	Asynchronous or synchronous reset supported. The Output Register (above) must be enabled first. Reset release requires that reset assertion be asynchronous.
Initialization	Select how to initialize the memory: all zeros, all ones, or from a memory file (see below). If you select Memory File, you may allow updates of the initialization file that is stored in the user flash memory (UFM) (not available with ECP5 and MachXO3L).
Memory File	Allows you to select a memory initialization (.mem) file for the module. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Allow update of initialization file stored in UFM	If you select Memory File (above), you may allow updates of the initialization file that is stored in the user flash memory (UFM). (Not available with ECP5 and MachXO3L)
Enable ECC	Allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths greater than 64 bits.
Pipeline Stages for Q and Error Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.

Table 269: RAM_DP Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2

Feature	Description
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1290, *Memory Usage Guide for MachXO3L Devices*

RAM_DP Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)

A pseudo-dual-port RAM that provides a variety of RAM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

Table 270: RAM_DP Dialog Box

Feature	Description
Specify the Size of RAM_DP	Both Read and Write port width can be specified. Enter number of address locations times data width. The two ports can be of different widths. In such cases, the memory initialization file corresponds to the Write Port. For example, for a RAM_DP with Write Port = 1024x36 and Read Port = 2048x18, the initialization file should be of size 1024x36.
Provide Byte Enables	Allows for byte enables at a size of 8 or 9 bytes. Choose the byte size from the drop-down menu. (Not available with LatticeEC, LatticeECP, LatticeXP, or MachXO.)
Enable Output Register	When selected, output register is enabled (separate controls for Read Port and Write Port).
Reset Mode	Asynchronous or synchronous reset supported. (Not available with LatticeECP3.)
Optimization	Select whether to optimize the module for minimum area or maximum speed.

Table 270: RAM_DP Dialog Box (Continued)

Feature	Description
Memory File	Allows you to select a memory initialization (.mem) file for the module. If a memory file is not used, all the RAM contents are initialized to 0. See "Creating a Memory Initialization File with Memory Generator" on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Enable ECC	Allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths greater than 64 bits.
Pipeline Stages for Q and Error Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with FPGA Memories" on page 1162
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*

pmi_ram_dp

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp.

Port Description

Input/ Output	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_wr_addr_width - 1):0
I	RdAddress	Bus	(pmi_rd_addr_width - 1):0
I	Data	Bus	(pmi_wr_data_width - 1):0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_rd_data_width - 1):0

Verilog pmi_ram_dp Definition

```
module pmi_ram_dp
  #(parameter pmi_wr_addr_depth = 512,
    parameter pmi_wr_addr_width = 9,
    parameter pmi_wr_data_width = 18,
    parameter pmi_rd_addr_depth = 512,
    parameter pmi_rd_addr_width = 9,
    parameter pmi_rd_data_width = 18,
    parameter pmi_regmode = "reg",
    parameter pmi_gsr = "disable",
    parameter pmi_resetmode = "sync",
    parameter pmi_optimization = "speed",
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_ram_dp")

    (input [(pmi_wr_data_width-1):0] Data,
     input [(pmi_wr_addr_width-1):0] WrAddress,
     input [(pmi_rd_addr_width-1):0] RdAddress,
     input  WrClock,
     input  RdClock,
     input  WrClockEn,
     input  RdClockEn,
     input  WE,
     input  Reset,
     output [(pmi_rd_data_width-1):0]  Q);/*synthesis
syn_black_box */

endmodule // pmi_ram_dp
```

VHDL pmi_ram_dp Definition

```
component pmi_ram_dp is
     generic (
```

```
            pmi_wr_addr_depth : integer := 512;
            pmi_wr_addr_width : integer := 9;
            pmi_wr_data_width : integer := 18;
            pmi_rd_addr_depth : integer := 512;
            pmi_rd_addr_width : integer := 9;
            pmi_rd_data_width : integer := 18;
            pmi_regmode : string := "reg";
            pmi_gsr : string := "disable";
            pmi_resetmode : string := "sync";
            pmi_optimization : string := "speed";
            pmi_init_file : string := "none";
            pmi_init_file_format : string := "binary";
            pmi_family : string := "EC";
            module_type : string := "pmi_ram_dp"
        );
        port (
         Data : in std_logic_vector((pmi_wr_data_width-1) downto
0);
         WrAddress : in std_logic_vector((pmi_wr_addr_width-1)
downto 0);
         RdAddress : in std_logic_vector((pmi_rd_addr_width-1)
downto 0);
         WrClock: in std_logic;
         RdClock: in std_logic;
         WrClockEn: in std_logic;
         RdClockEn: in std_logic;
         WE: in std_logic;
         Reset: in std_logic;
         Q : out std_logic_vector((pmi_rd_data_width-1) downto 0)
       );
      end component pmi_ram_dp;
```

Parameter Names and Values

The following table contains the pmi_ram_dp module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_rd_addr_depth	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_rd_addr_width	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_rd_data_width	1 to 256	18	
pmi_wr_addr_depth	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_wr_addr_width	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_wr_data_width	1 to 256	18	
pmi_regmode	“reg” | “noreg”	“reg”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_gsr	“enable”	“disable”	“disable”	
pmi_resetmode	“async”	“sync”	“sync”	The “async” option is not valid for LatticeECP3 devices.
pmi_init_file	<string>	“none”	
pmi_init_file_format	“binary”	“hex”	“binary”	
pmi_family	“EC”	“ECP”	“ECP2”	“ECP2M”	“ECP3”	“ECP5U”	“ECP5UM”	“SC”	“SCM”	“XP”	“XP2”	“XO”	“XO2”	“XO3L”	“LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

pmi_ram_dp_be

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp_be, which is similar to pmi_ram_dp but with a byte enable.

Port Description

Input/ Output	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_wr_addr_width - 1):0
I	RdAddress	Bus	(pmi_rd_addr_width - 1):0
I	Data	Bus	(pmi_wr_data_width - 1):0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
I	ByteEn	Bit	((pmi_wr_data_width + pmi_byte_size-1) / pmi_byte_size-1):0
O	Q	Bus	(pmi_rd_data_width - 1):0

Verilog pmi_ram_dp_be Definition

```
module pmi_ram_dp_be
 #(parameter pmi_wr_addr_depth = 512,
   parameter pmi_wr_addr_width = 9,
   parameter pmi_wr_data_width = 18,
   parameter pmi_rd_addr_depth = 512,
   parameter pmi_rd_addr_width = 9,
   parameter pmi_rd_data_width = 18,
   parameter pmi_regmode = "reg",
   parameter pmi_gsr = "disable",
   parameter pmi_resetmode = "sync",
   parameter pmi_optimization = "speed",
   parameter pmi_init_file = "none",
   parameter pmi_init_file_format = "binary",
   parameter pmi_byte_size = 9,
   parameter pmi_family = "ECP2",
   parameter module_type = "pmi_ram_dp_be")

   (input [(pmi_wr_data_width-1):0] Data,
    input [(pmi_wr_addr_width-1):0] WrAddress,
    input [(pmi_rd_addr_width-1):0] RdAddress,
    input  WrClock,
    input  RdClock,
    input  WrClockEn,
    input  RdClockEn,
    input  WE,
    input  Reset,
    input [((pmi_wr_data_width+pmi_byte_size-1)/pmi_byte_size-
1):0] ByteEn,
```

```
    output [(pmi_rd_data_width-1):0]  Q) /*synthesis
syn_black_box*/;

endmodule // pmi_ram_dp
```

VHDL pmi_ram_dp_be Definition

```
component pmi_ram_dp_be is
     generic (
       pmi_wr_addr_depth : integer := 512;
       pmi_wr_addr_width : integer := 9;
       pmi_wr_data_width : integer := 18;
       pmi_rd_addr_depth : integer := 512;
       pmi_rd_addr_width : integer := 9;
       pmi_rd_data_width : integer := 18;
       pmi_regmode : string := "reg";
       pmi_gsr : string := "disable";
       pmi_resetmode : string := "sync";
       pmi_optimization : string := "speed";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_byte_size : integer := 9;
       pmi_family : string := "ECP2";
       module_type : string := "pmi_ram_dp_be"
    );
    port (
     Data : in std_logic_vector((pmi_wr_data_width-1) downto
0);
     WrAddress : in std_logic_vector((pmi_wr_addr_width-1)
downto 0);
     RdAddress : in std_logic_vector((pmi_rd_addr_width-1)
downto 0);
     WrClock: in std_logic;
     RdClock: in std_logic;
     WrClockEn: in std_logic;
     RdClockEn: in std_logic;
     WE: in std_logic;
     Reset: in std_logic;
     ByteEn : in
std_logic_vector(((pmi_wr_data_width+pmi_byte_size-1)/
pmi_byte_size-1) downto 0);
     Q : out std_logic_vector((pmi_rd_data_width-1) downto 0)
   );
  end component pmi_ram_dp_be;
```

Parameter Names and Values

The following table contains the pmi_ram_dp_be module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_rd_addr_depth	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_rd_addr_width	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_rd_data_width	1 to 256	18	
pmi_wr_addr_depth	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_wr_addr_width	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_wr_data_width	1 to 256	18	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_gsr	“enable” | “disable”	“disable”	
pmi_resetmode	“async” | “sync”	“sync”	The “async” option is not valid for LatticeECP3 devices.
pmi_optimization	“area”| “speed”	“speed”	
pmi_init_file	<string>	“none”	
pmi_init_file_format	“binary” | “hex”	“binary”	
pmi_byte_size	8 | 9	9	
pmi_family	“ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP2” | “XO2” | “XO3L”	“ECP2”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

RAM_DP_TRUE

A true dual-port RAM that provides a variety of RAM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

The options vary with the device family. See:

- “RAM_DP_TRUE Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)” on page 1615
- “RAM_DP_TRUE Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)” on page 1617

See Also

- “Use PMI or IPexpress?” on page 191

- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with FPGA Memories" on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1306, *CrossLink Memory Usage Guide*

RAM_DP_TRUE Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)

The module dialog box has two tabs:

- Configuration, below
- Advanced, see Table 272 on page 1616

Table 271: RAM_DP_TRUE Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2—Configuration Tab

Feature	Description
Specify the Size of the RAM_DP_TRUE	Both Read and Write port widths can be specified. Enter the number of address locations and the data width. The size of the memory initialization file corresponds to the port A. For example, for a RAM_DP_TRUE with port A = 1024x36 and port B = 2048x18, the initialization file should be of size 1024x36.
Enable Output Register	When selected, the output registered is enabled (separate controls for ports A and B).
Enable Output ClockEn	Adds an output clock enable port. This is an input port. Requires that the Output Register (above) be enabled first.
Provide Byte Enables	Allows for byte enables at a size of 8 or 9 bytes. Choose the byte size from the drop-down menu.
Optimization	Select whether to optimize the module for minimum area or maximum speed.
Reset Mode	Asynchronous or synchronous reset supported. The Output Register (above) must be enabled first. Reset release requires that reset assertion be asynchronous.

Table 271: RAM_DP_TRUE Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2—Configuration Tab (Continued)

Feature	Description
Initialization	Select how to initialize the memory: all zeros, all ones, or from a memory file (see below). If you select Memory File, you may allow updates of the initialization file that is stored in the user flash memory (UFM) (not available with ECP5 and MachXO3L).
Memory File	Allows you to select a memory initialization (.mem) file for the module. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Allow update of initialization file stored in UFM	If you select Memory File (above), you may allow updates of the initialization file that is stored in the user flash memory (UFM). (Not available with ECP5 and MachXO3L)
Enable ECC	Allows error correction of single errors and detection of 2-bit errors. Ports A and B must have the same data width and the data width cannot be greater than 64 bits.
Pipeline Stages for Q and Error Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 272: RAM_DP_TRUE Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2—Advanced Tab

Feature	Description
Port A, B Write Mode	Supports: ▶ Normal ▶ Write Through ▶ Read Before Write. A memory configuration can only use read-before-write if its data width is a multiple of 9. Set Data Width in the Configuration tab to match this requirement.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with FPGA Memories" on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1290, *Memory Usage Guide for MachXO3L Devices*

RAM_DP_TRUE Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)

The module dialog box has two tabs:

- Configuration, below
- Advanced, see Table 274 on page 1618

Table 273: RAM_DP_TRUE Dialog Box—Configuration Tab

Feature	Description
Specify the Size of RAM_DP_TRUE	Both Read and Write port width can be specified. Enter the number of address locations and the data width. The size of the memory initialization file corresponds to the port A. For example, for a RAM_DP_TRUE with port A = 1024x36 and port B = 2048x18, the initialization file should be of size 1024x36.
Enable Output Register	When selected, output register is enabled (separate controls for port A and B).
Provide Byte Enables	Allows for byte enables at a size of 8 or 9 bytes. Choose the byte size from the drop-down menu. (Not available with LatticeEC, LatticeECP, LatticeXP, or MachXO.)
Reset Mode	Asynchronous or synchronous reset supported. (Not available with LatticeECP3.)
Optimization	Select whether to optimize the module for minimum area or maximum speed.
Memory File	Allows you to select a memory initialization (.mem) file for the module. If a memory file is not used, all the RAM contents are initialized to 0. See "Creating a Memory Initialization File with Memory Generator" on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Enable ECC	Allows error correction of single errors and detection of 2-bit errors.
Pipeline Stages for Q and Error Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.
Bus Ordering Style	Specifies whether the numbers should begin at the high end ("Big Endian"- BusA[7 to 0]) or at the low end ("Little Endian"-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.

Table 273: RAM_DP_TRUE Dialog Box—Configuration Tab (Continued)

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 274: RAM_DP_TRUE Dialog Box—Advanced Tab

Feature	Description
Port A, B Write Mode	Supports: ▶ Normal ▶ Write Through ▶ Read before Write. A memory configuration can only use read-before-write if its data width is a multiple of 9. Set Data Width in the Configuration tab to match this requirement. (Not available with LatticeECP2/M, LatticeSC/M, and LatticeXP2.)
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*

pmi_ram_dp_true

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp_true.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataInA	Bus	(pmi_data_width_a - 1):0
I	DataInB	Bus	(pmi_data_width_b - 1):0
I	AddressA	Bus	(pmi_addr_width_a - 1):0
I	AddressB	Bus	(pmi_addr_width_b - 1):0
I	ClockA	Bit	N/A
I	ClockB	Bit	N/A
I	ClockEnA	Bit	N/A
I	ClockEnB	Bit	N/A
I	WrA	Bit	N/A
I	WrB	Bit	N/A
I	ResetA	Bit	N/A
I	ResetB	Bit	N/A
O	QA	Bus	(pmi_data_width_a - 1):0
O	QB	Bus	(pmi_data_width_b - 1):0

Verilog pmi_ram_dp_true Definition

```
module pmi_ram_dp_true
  #(parameter pmi_addr_depth_a = 512,
    parameter pmi_addr_width_a = 9,
    parameter pmi_data_width_a = 18,
    parameter pmi_addr_depth_b = 512,
    parameter pmi_addr_width_b = 9,
    parameter pmi_data_width_b = 18,
    parameter pmi_regmode_a = "reg",
    parameter pmi_regmode_b = "reg",
    parameter pmi_gsr = "disable",
    parameter pmi_resetmode = "sync",
    parameter pmi_optimization = "speed",
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_write_mode_a = "normal",
    parameter pmi_write_mode_b = "normal",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_ram_dp_true")

    (input [(pmi_data_width_a-1):0]DataInA,
     input [(pmi_data_width_b-1):0]DataInB,
     input [(pmi_addr_width_a-1):0] AddressA,
```

```
     input [(pmi_addr_width_b-1):0] AddressB,
     input  ClockA,
     input  ClockB,
     input  ClockEnA,
     input  ClockEnB,
     input  WrA,
     input  WrB,
     input  ResetA,
     input  ResetB,
     output [(pmi_data_width_a-1):0]  QA,
     output [(pmi_data_width_b-1):0]  QB)/*synthesis
syn_black_box */;

endmodule // pmi_ram_dp_true
```

VHDL pmi_ram_dp_true Definition

```
component pmi_ram_dp_true is
     generic (
       pmi_addr_depth_a : integer := 512;
       pmi_addr_width_a : integer := 9;
       pmi_data_width_a : integer := 18;
       pmi_addr_depth_b : integer := 512;
       pmi_addr_width_b : integer := 9;
       pmi_data_width_b : integer := 18;
       pmi_regmode_a : string := "reg";
       pmi_regmode_b : string := "reg";
       pmi_gsr : string := "disable";
       pmi_resetmode : string := "sync";
       pmi_optimization : string := "speed";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_write_mode_a : string := "normal";
       pmi_write_mode_b : string := "normal";
       pmi_family : string := "EC";
       module_type : string := "pmi_ram_dp_true"
    );
    port (
     DataInA : in std_logic_vector((pmi_data_width_a-1) downto
0);
     DataInB : in std_logic_vector((pmi_data_width_b-1) downto
0);
     AddressA : in std_logic_vector((pmi_addr_width_a-1) downto
0);
     AddressB : in std_logic_vector((pmi_addr_width_b-1) downto
0);
     ClockA: in std_logic;
     ClockB: in std_logic;
     ClockEnA: in std_logic;
     ClockEnB: in std_logic;
     WrA: in std_logic;
     WrB: in std_logic;
     ResetA: in std_logic;
     ResetB: in std_logic;
     QA : out std_logic_vector((pmi_data_width_a-1) downto 0);
     QB : out std_logic_vector((pmi_data_width_b-1) downto 0)
   );
  end component pmi_ram_dp_true;
```

Parameter Names and Values

The following table contains the pmi_ram_dp_true module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_addr_depth_a	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_addr_width_a	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_data_width_a	1 to 256	18	
pmi_addr_depth_b	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_addr_width_b	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_data_width_b	1 to 256	18	
pmi_regmode_a	“reg” | “noreg”	“reg”	
pmi_regmode_b	“reg” | “noreg”	“reg”	
pmi_gsr	“enable” | “disable”	“disable”	
pmi_resetmode	“async” | “sync”	“sync”	The “async” option is not valid for LatticeECP3 devices.
pmi_init_file	<string>	“none”	
pmi_init_file_format	“binary” | “hex”	“binary”	
pmi_write_mode_a	“normal” | “writethrough” | “readbeforewrite”	“normal”	The “readbeforewrite” option is not supported for the LatticeECP2/M, LatticeSC/M, and LatticeXP device families.
pmi_write_mode_b	“normal” | “writethrough” | “readbeforewrite”	“normal”	The “readbeforewrite” option is not supported for the LatticeECP2/M, LatticeSC/M, and LatticeXP device families.
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “| “XO2” | “XO3L” | “LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

pmi_ram_dp_true_be

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dp_true_be, which is similar to pmi_ram_dp_true but with a byte enable.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataInA	Bus	(pmi_data_width_a - 1):0
I	DataInB	Bus	(pmi_data_width_b - 1):0
I	AddressA	Bus	(pmi_addr_width_a - 1):0
I	AddressB	Bus	(pmi_addr_width_b - 1):0
I	ClockA	Bit	N/A
I	ClockB	Bit	N/A
I	ClockEnA	Bit	N/A
I	ClockEnB	Bit	N/A
I	WrA	Bit	N/A
I	WrB	Bit	N/A
I	ResetA	Bit	N/A
I	ResetB	Bit	N/A
I	ByteEnA	Bit	((pmi_data_width_a + pmi_byte_size-1) / pmi_byte_size-1):0
I	ByteEnB	Bit	((pmi_data_width_b + pmi_byte_size-1) / pmi_byte_size-1):0
O	QA	Bus	(pmi_data_width_a - 1):0
O	QB	Bus	(pmi_data_width_b - 1):0

Verilog pmi_ram_dp_true_be Definition

```
module pmi_ram_dp_true_be
 #(parameter pmi_addr_depth_a = 512,
   parameter pmi_addr_width_a = 9,
   parameter pmi_data_width_a = 18,
   parameter pmi_addr_depth_b = 512,
   parameter pmi_addr_width_b = 9,
   parameter pmi_data_width_b = 18,
   parameter pmi_regmode_a = "reg",
   parameter pmi_regmode_b = "reg",
   parameter pmi_gsr = "disable",
   parameter pmi_resetmode = "sync",
   parameter pmi_optimization = "speed",
   parameter pmi_init_file = "none",
```

```
   parameter pmi_init_file_format = "binary",
   parameter pmi_write_mode_a = "normal",
   parameter pmi_write_mode_b = "normal",
   parameter pmi_byte_size = 9,
   parameter pmi_family = "ECP2",
   parameter module_type = "pmi_ram_dp_true_be")

  (input [(pmi_data_width_a-1):0]DataInA,
   input [(pmi_data_width_b-1):0]DataInB,
   input [(pmi_addr_width_a-1):0] AddressA,
   input [(pmi_addr_width_b-1):0] AddressB,
   input  ClockA,
   input  ClockB,
   input  ClockEnA,
   input  ClockEnB,
   input  WrA,
   input  WrB,
   input  ResetA,
   input  ResetB,
   input [((pmi_data_width_a+pmi_byte_size-1)/pmi_byte_size-
1):0] ByteEnA,
   input [((pmi_data_width_b+pmi_byte_size-1)/pmi_byte_size-
1):0] ByteEnB,
   output [(pmi_data_width_a-1):0]  QA,
   output [(pmi_data_width_b-1):0]  QB) /*synthesis
syn_black_box*/;

endmodule // pmi_ram_dp_true
```

VHDL pmi_ram_dp_true_be Definition

```
component pmi_ram_dp_true_be is
     generic (
       pmi_addr_depth_a : integer := 512;
       pmi_addr_width_a : integer := 9;
       pmi_data_width_a : integer := 18;
       pmi_addr_depth_b : integer := 512;
       pmi_addr_width_b : integer := 9;
       pmi_data_width_b : integer := 18;
       pmi_regmode_a : string := "reg";
       pmi_regmode_b : string := "reg";
       pmi_gsr : string := "disable";
       pmi_resetmode : string := "sync";
       pmi_optimization : string := "speed";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_write_mode_a : string := "normal";
       pmi_write_mode_b : string := "normal";
       pmi_byte_size : integer := 9;
       pmi_family : string := "ECP2";
       module_type : string := "pmi_ram_dp_true_be"
    );
    port (
     DataInA : in std_logic_vector((pmi_data_width_a-1) downto
0);
     DataInB : in std_logic_vector((pmi_data_width_b-1) downto
0);
     AddressA : in std_logic_vector((pmi_addr_width_a-1) downto
0);
```

```
     AddressB : in std_logic_vector((pmi_addr_width_b-1) downto
0);
     ClockA: in std_logic;
     ClockB: in std_logic;
     ClockEnA: in std_logic;
     ClockEnB: in std_logic;
     WrA: in std_logic;
     WrB: in std_logic;
     ResetA: in std_logic;
     ResetB: in std_logic;
     ByteEnA : in
std_logic_vector(((pmi_data_width_a+pmi_byte_size-1)/
pmi_byte_size-1) downto 0);
     ByteEnB : in
std_logic_vector(((pmi_data_width_b+pmi_byte_size-1)/
pmi_byte_size-1) downto 0);
     QA : out std_logic_vector((pmi_data_width_a-1) downto 0);
     QB : out std_logic_vector((pmi_data_width_b-1) downto 0)
   );
end component pmi_ram_dp_true_be;
```

Parameter Names and Values

The following table contains the pmi_ram_dp_true module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_addr_depth_a	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_addr_width_a	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_data_width_a	1 to 256	18	
pmi_addr_depth_b	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_addr_width_b	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_data_width_b	1 to 256	18	
pmi_regmode_a	“reg” | “noreg”	“reg”	
pmi_regmode_b	“reg” | “noreg”	“reg”	
pmi_gsr	“enable” | “disable”	“disable”	
pmi_resetmode	“async” | “sync”	“sync”	The “async” option is not valid for LatticeECP3 devices.
pmi_optimization	“area” | “speed”	“speed”	
pmi_init_file	<string>	“none”	

Parameter Name	Acceptable Values	Default Value	Notes
pmi_init_file_format	“binary”	“hex”	“binary”	
pmi_write_mode_a	“normal”	“writethrough”	“readbeforewrite”	“normal”	The “readbeforewrite” option is not supported for the LatticeECP2/M, LatticeSC/M, and LatticeXP device families.
pmi_write_mode_b	“normal”	“writethrough”	“readbeforewrite”	“normal”	The “readbeforewrite” option is not supported for the LatticeECP2/M, LatticeSC/M, and LatticeXP device families.
pmi_byte_size	8	9	9	
pmi_family	“ECP2”	“ECP2M”	“ECP3”	“ECP5U”	“ECP5UM”	“SC”	“SCM”	“XP2”	“XO2”| “XO2”	“XO3L”	“ECP2”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

RAM_DQ

A RAM that can support registered and unregistered outputs. The module provides a variety of RAM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

The options vary with the device family. See:

- “RAM_DQ Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)” on page 1626
- “RAM_DQ Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)” on page 1627

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*

- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*
- TN1306, *CrossLink Memory Usage Guide*

RAM_DQ Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)

The module dialog box has two tabs:

- Configuration, below
- Advanced, see Table 276 on page 1627

Table 275: RAM_DQ Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2—Configuration Tab

Feature	Description
Specify the size of the RAM_DQ	Specify the number of address locations and the data width.
Provide Byte Enables	Allows for byte enables at a size of 8 or 9 bytes. Choose the byte size from the drop-down menu. The data width must be at least 9 bits.
Enable Output Register	When selected, output registered is enabled.
Enable Output ClockEn	Adds an output clock enable port. This is an input port. Requires that the Output Register (above) be enabled first.
Optimization	Select whether to optimize the module for minimum area or maximum speed.
Reset Mode	Asynchronous or synchronous reset supported. The Output Register (above) must be enabled first. Reset release requires that reset assertion be asynchronous.
Initialization	Select how to initialize the memory: all zeros, all ones, or from a memory file (see below). If you select Memory File, you may allow updates of the initialization file that is stored in the user flash memory (UFM) (not available with ECP5 or MachXO3L).
Memory File	Allows you to select a memory initialization (.mem) file for the module. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Allow update of initialization file stored in UFM	If you select Memory File (above), you may allow updates of the initialization file that is stored in the user flash memory (UFM). (Not available with ECP5 or MachXO3L)
Enable ECC	Allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths greater than 64 bits.
Pipeline Stages for Q and Error Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.

Table 275: RAM_DQ Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2—Configuration Tab (Continued)

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 276: RAM_DQ Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2—Advanced Tab

Feature	Description
Write Mode	Supports: ▶ Normal ▶ Write Through ▶ Read before Write. A memory configuration can only use read-before-write if its data width is a multiple of 9. Set Data Width in the Configuration tab to match this requirement.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1290, *Memory Usage Guide for MachXO3L Devices*

RAM_DQ Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)

The module dialog box has two tabs:

- Configuration, below
- Advanced, see Table 278 on page 1628

Table 277: RAM_DQ Dialog Box—Configuration Tab

Feature	Description
Specify the size of RAM_DQ	Specify the number of address locations and the data width.
Enable Output Register	When selected, output register is enabled.
Provide Byte Enables	Allows for byte enables at a size of 8 or 9 bytes. Choose the byte size from the drop-down menu. The data width must be at least 9 bits. (Not available with LatticeEC, LatticeECP, LatticeXP, or MachXO.)
Reset Mode	Asynchronous or synchronous reset supported. (Not available with LatticeECP3.)
Optimization	Select whether to optimize the module for minimum area or maximum speed.
Memory File	Allows you to select a memory initialization (.mem) file for the module. If a memory file is not used, all the RAM contents are initialized to 0. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Enable ECC (not supported for Data Width > 64)	Allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths greater than 64 bits.
Pipeline Stages for Q and ERROR Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 278: RAM_DQ Dialog Box—Advanced Tab

Feature	Description
Write Mode	Supports: - Normal - Write Through - Read before Write. A memory configuration can only use read-before-write if its data width is a multiple of 9. Set Data Width in the Configuration tab to match this requirement. (Not available with LatticeECP2/M, LatticeSC/M, and LatticeXP2.)
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.

Table 278: RAM_DQ Dialog Box—Advanced Tab (Continued)

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Use PMI or IPexpress?" on page 191
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with FPGA Memories" on page 1162
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*

pmi_ram_dq

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dq.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width - 1):0
I	Address	Bus	(pmi_addr_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_ram_dq Definition

```
module pmi_ram_dq
  #(parameter pmi_addr_depth = 512,
```

```
    parameter pmi_addr_width = 9,
    parameter pmi_data_width = 18,
    parameter pmi_regmode = "reg",
    parameter pmi_gsr = "disable",
    parameter pmi_resetmode = "sync",
    parameter pmi_optimization = "speed",
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_write_mode = "normal",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_ram_dq")

    (input [(pmi_data_width-1):0]Data,
     input [(pmi_addr_width-1):0] Address,
     input  Clock,
     input  ClockEn,
     input  WE,
     input  Reset,
     output [(pmi_data_width-1):0]  Q)/*synthesis
syn_black_box*/;

endmodule // pmi_ram_dq
```

VHDL pmi_ram_dq Definition

```
component pmi_ram_dq is
     generic (
       pmi_addr_depth : integer := 512;
       pmi_addr_width : integer := 9;
       pmi_data_width : integer := 18;
       pmi_regmode : string := "reg";
       pmi_gsr : string := "disable";
       pmi_resetmode : string := "sync";
       pmi_optimization : string := "speed";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_write_mode : string := "normal";
       pmi_family : string := "EC";
       module_type : string := "pmi_ram_dq"
    );
    port (
     Data : in std_logic_vector((pmi_data_width-1) downto 0);
     Address : in std_logic_vector((pmi_addr_width-1) downto
0);
     Clock: in std_logic;
     ClockEn: in std_logic;
     WE: in std_logic;
     Reset: in std_logic;
     Q : out std_logic_vector((pmi_data_width-1) downto 0)
   );
  end component pmi_ram_dq;
```

Parameter Names and Values

The following table contains the pmi_ram_dq module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_addr_depth	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_addr_width	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_data_width	1 to 256	18	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_gsr	“enable” | “disable”	“disable”	
pmi_resetmode	“async” | “sync”	“sync”	The “async” option is not valid for LatticeECP3 devices.
pmi_init_file	<string>	“none”	
pmi_init_file_format	“binary” | “hex”	“binary”	
pmi_write_mode	“normal” | “writethrough” | “readbeforewrite”	“normal”	The “readbeforewrite” option is not supported for the LatticeECP2/M, LatticeSC/M, and LatticeXP device families.
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” | “LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

pmi_ram_dq_be

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_ram_dq_be, which is similar to pmi_ram_dq with the addition of a byte enable.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width - 1):0
I	Address	Bus	(pmi_addr_width - 1):0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WE	Bit	N/A
I	ByteEn	Bus	((pmi_data_width + pmi_byte_size-1) / pmi_byte_size-1):0
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_ram_dqbe Definition

```
module pmi_ram_dq_be
 #(parameter pmi_addr_depth = 512,
   parameter pmi_addr_width = 9,
   parameter pmi_data_width = 18,
   parameter pmi_regmode = "reg",
   parameter pmi_gsr = "disable",
   parameter pmi_resetmode = "sync",
   parameter pmi_optimization = "speed",
   parameter pmi_init_file = "none",
   parameter pmi_init_file_format = "binary",
   parameter pmi_write_mode = "normal",
   parameter pmi_byte_size = 9,
   parameter pmi_family = "ECP2",
   parameter module_type = "pmi_ram_dq_be")

  (input [(pmi_data_width-1):0]Data,
   input [(pmi_addr_width-1):0] Address,
   input  Clock,
   input  ClockEn,
   input  WE,
   input  Reset,
  input [((pmi_data_width+pmi_byte_size-1)/pmi_byte_size-1):0]
ByteEn,
   output [(pmi_data_width-1):0]  Q) /*synthesis
syn_black_box*/;

endmodule // pmi_ram_dq_be
```

VHDL pmi_ram_dq_be Definition

```
component pmi_ram_dq_be is
     generic (
       pmi_addr_depth : integer := 512;
       pmi_addr_width : integer := 9;
       pmi_data_width : integer := 18;
       pmi_regmode : string := "reg";
```

```
        pmi_gsr : string := "disable";
        pmi_resetmode : string := "sync";
        pmi_optimization : string := "speed";
        pmi_init_file : string := "none";
        pmi_init_file_format : string := "binary";
        pmi_write_mode : string := "normal";
        pmi_byte_size : integer := 9;
        pmi_family : string := "ECP2";
        module_type : string := "pmi_ram_dq_be"
    );
    port (
     Data : in std_logic_vector((pmi_data_width-1) downto 0);
     Address : in std_logic_vector((pmi_addr_width-1) downto
0);
     Clock: in std_logic;
     ClockEn: in std_logic;
     WE: in std_logic;
     Reset: in std_logic;
     ByteEn : in
std_logic_vector(((pmi_data_width+pmi_byte_size-1)/
pmi_byte_size-1) downto 0);
     Q : out std_logic_vector((pmi_data_width-1) downto 0)
   );
end component pmi_ram_dq_be;
```

Parameter Names and Values

The following table contains the pmi_ram_dq_be module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_addr_depth	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_addr_width	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_data_width	1 to 256	18	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_gsr	“enable” | “disable”	“disable”	
pmi_resetmode	“async” | “sync”	“sync”	The “async” option is not valid for LatticeECP3 devices.
pmi_optimization	“area” | “speed”	“speed”	
pmi_init_file	<string>	“none”	
pmi_init_file_format	“binary” | “hex”	“binary”	
pmi_write_mode	“normal” | “writethrough” | “readbeforewrite”	“normal”	The “readbeforewrite” option is not supported for the LatticeECP2/M, LatticeSC/M, and LatticeXP device families.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_byte_size	8 | 9	9	
pmi_family	“ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “SC” | “SCM” | “XP2” | “XO2” | “XO3L”	“ECP2”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

ROM

A ROM that provides a variety of ROM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

The options vary with the device family. See:

- “ROM Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)” on page 1635
- “ROM Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)” on page 1636

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1179, *LatticeECP3 Memory Usage Guide*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1290, *Memory Usage Guide for MachXO3L Devices*

ROM Options (ECP5, MachXO2, MachXO3L, Platform Manager 2)

A ROM that provides a variety of ROM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

Table 279: ROM Dialog Box for ECP5, MachXO2, MachXO3L, and Platform Manager 2

Feature	Description
Specify the size of the ROM	Specify the number of address locations and the data width.
Enable Output Register	When selected, output registered is enabled.
Optimization	Select whether to optimize the module for minimum area or maximum speed.
Reset Mode	Asynchronous or synchronous reset supported. The Output Register (above) must be enabled first. Reset release requires that reset assertion be asynchronous.
Memory File	Allows you to select a memory initialization (.mem) file for the module. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Allow update of initialization file stored in UFM	Allow updates of the initialization file that is stored in the user flash memory (UFM). (Not available with ECP5 and MachXO3L)
Enable ECC	Allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths greater than 64 bits.
Pipeline Stages for Q and ERROR Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1264, *ECP5 Memory Usage Guide*
- TN1290, *Memory Usage Guide for MachXO3L Devices*

ROM Options (LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeSC/M, LatticeXP, LatticeXP2, MachXO)

A ROM that provides a variety of ROM organizations and is implemented as sysMEM EBR memory. Ports, properties, and functionality are described in detail in the technical notes listed in the “See Also” section below.

Table 280: ROM Dialog Box

Feature	Description
Specify the size of the ROM	Specify the number of address locations and the data width.
Enable Output Register	When selected, output register is enabled.
Reset Mode	Asynchronous or synchronous reset supported. (Not available with LatticeECP3.)
Optimization	Select whether to optimize the module for minimum area or maximum speed.
Memory File	Allows you to select a memory initialization (.mem) file for the module. See “Creating a Memory Initialization File with Memory Generator” on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Enable ECC	Allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths greater than 64 bits.
Pipeline Stages for Q and ERROR Outputs	If ECC is enabled, the output can be pipelined. Choose the number of stages from the drop-down menu.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195
- “How to Design with FPGA Memories” on page 1162
- TN1051, *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices*
- TN1094, *On-Chip Memory Usage Guide for LatticeSC Devices*
- TN1104, *LatticeECP2/M Memory Usage Guide*
- TN1092, *Memory Usage Guide for MachXO2 Devices*
- TN1137, *LatticeXP2 Memory Usage Guide*

pmi_rom

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_rom.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width - 1):0
I	OutClock	Bit	N/A
I	OutClockEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width - 1):0

Verilog pmi_rom Definition

```
module pmi_rom
  #(parameter pmi_addr_depth = 512,
    parameter pmi_addr_width = 9,
    parameter pmi_data_width = 8,
    parameter pmi_regmode = "reg",
    parameter pmi_gsr = "disable",
    parameter pmi_resetmode = "sync",
    parameter pmi_optimization = "speed",
    parameter pmi_init_file = "none",
    parameter pmi_init_file_format = "binary",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_rom")

    (input [(pmi_addr_width-1):0]Address,
     input OutClock,
     input OutClockEn,
     input Reset,
     output [(pmi_data_width-1):0] Q)/*synthesis
syn_black_box*/;

endmodule // pmi_rom
```

VHDL pmi_rom Definition

```
component pmi_rom is
     generic (
       pmi_addr_depth : integer := 512;
       pmi_addr_width : integer := 9;
       pmi_data_width : integer := 8;
       pmi_regmode : string := "reg";
       pmi_gsr : string := "disable";
       pmi_resetmode : string := "sync";
       pmi_optimization : string := "speed";
       pmi_init_file : string := "none";
       pmi_init_file_format : string := "binary";
       pmi_family : string := "EC";
       module_type : string := "pmi_rom"
    );
```

```
    port (
     Address : in std_logic_vector((pmi_addr_width-1) downto
0);
     OutClock: in std_logic;
     OutClockEn: in std_logic;
     Reset: in std_logic;
     Q : out std_logic_vector((pmi_data_width-1) downto 0)
   );
  end component pmi_rom;
```

Parameter Names and Values

The following table contains the pmi_rom module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_addr_depth	2 to 65536	512	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_addr_width	1 to 16	9	The LatticeECP2 and LatticeSC/M device families read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
pmi_data_width	1 to 256	8	
pmi_regmode	“reg” | “noreg”	“reg”	
pmi_gsr	“enable” | “disable”	“disable”	
pmi_resetmode	“async” | “sync”	“sync”	The “async” option is not valid for LatticeECP3 devices.
pmi_init_file	<string>	“none”	
pmi_init_file_format	“binary” | “hex”	“binary”	
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “ XO2” | “XO3L” | “LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

SDR

Single data-rate registers of specified width and mode. Although an SDR generated for a specific width can be used for other widths, it usually requires extra care in specifying the connection while instantiating.

The options vary with the device family. See:

- “SDR Options (ECP5)” on page 1639
- “SDR Options (LatticeECP3, MachXO2, MachXO3L, Platform Manager 2)” on page 1640
- “SDR Options (LatticeSC/M)” on page 1641

See Also

- “Generating a Module or IP with IPexpress” on page 195
- TN1265, *ECP5 High-Speed I/O Interface*
- TN1180, *LatticeECP3 High-Speed I/O Interface*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1281, *Implementing High-Speed Interfaces with MachXO3L Devices*

SDR Options (ECP5)

Table 281: SDR Dialog Box for ECP5

Feature	Description
Interface Type	Specify whether the interface is for transmit or receive.
Enable Tri-state Control	Available only with the Transmit interface type.
I/O Standard for this Interface	Choose an I/O standard.
Bus Width for this Interface	Specify the width of the bus in bits.
Clock Frequency for this Interface	Specify the frequency of the clock in megahertz.
Interface Bandwidth	Shows the resulting bandwidth in megabits/second. Bandwidth is the bus width multiplied by the clock frequency. This field is read-only.
Interface	Shows the interface in terms of primitives. Depends on the Interface Type above.
Clock Inversion	Specify whether or not to enable clock inversion. Available only with the receive interface type.
Data Path Delay	Choose a delay type. The delay types depend on the Interface Type. - Bypass means no delay. - Static sets a fixed delay. - Dynamic adds a data_direction, data_loadn, data_move, and data_cflag ports each of which is as wide as the Bus Width. - User Defined allows you to set a delay value in the Delay Value for User Defined field.
Delay Value for User Defined	If Data Path Delay is “User Defined,” specify the number of fine delay increments. Refer to the device datasheet to see the delay value for each increment.

Table 281: SDR Dialog Box for ECP5

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- TN1265, *ECP5 High-Speed I/O Interface*

SDR Options (LatticeECP3, MachXO2, MachXO3L, Platform Manager 2)

Table 282: SDR Dialog Box for LatticeECP3, MachXO2, MachXO3L, and Platform Manager 2

Feature	Description
Interface Type	Specify whether the interface is for transmit or receive.
I/O Standard for this Interface	Choose an I/O standard.
Bus Width for this Interface	Specify the width of the bus in bits.
Clock Frequency for this Interface	Specify the frequency of the clock in megahertz.
Bandwidth (calculated)	Shows the resulting bandwidth in megabits/second. Bandwidth is the bus width multiplied by the clock frequency. This field is read-only.
Interface	Shows the interface. Depends on the Interface Type above.
Clock Inversion	Specify whether to enable clock inversion.

Table 282: SDR Dialog Box for LatticeECP3, MachXO2, MachXO3L, and Platform Manager 2

Feature	Description
Data Path Delay	Choose a delay type. The delay types depend on the device. Available only with the receive interface type. LatticeECP3: ▶ Bypass means no delay. ▶ Dynamic adds a 4-bit control port, del[3:0]. The value on the control port specifies the number of fine delay increments. Each increment is 1/16 of the clock cycle. ▶ User Defined sets a fixed delay. See FDEL for User Defined to set the amount. ▶ Fixed Delay adds a 4-bit control port, del[3:0]. The value on the control port specifies the number of fine delay increments. Each increment is 1/16 of the clock cycle. MachXO2, MachXO3L, or Platform Manager 2: ▶ Bypass means no delay. ▶ SCLK_ZEROHOLD sets zero hold for the register. ▶ User Defined sets a fixed delay. See FDEL for User Defined to set the amount.
FDEL for User Defined	If Data Path Delay is “User Defined,” specify the number of fine delay increments. Refer to the device datasheet to see the delay value for each increment. Available only with the receive interface type.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- TN1180, *LatticeECP3 High-Speed I/O Interface*
- TN1203, *Implementing High-Speed Interfaces with MachXO2 Devices*
- TN1281, *Implementing High-Speed Interfaces with MachXO3L Devices*

SDR Options (LatticeSC/M)

The module dialog box has two tabs:

- Configuration, below
- Advanced, see Table 284 on page 1643

Table 283: SDR Dialog Box for LatticeSC/M—Configuration Tab

Feature	Description
Mode	Specify the Mode of the SDR. The SDR can be used as an input, output, tristate, or bidirectional. A tristate mode consists of two output SDRs. A bidirectional mode has one input and two output SDRs. Values: Input, Output, Bidirectional, Tristate
Bus Width	Specify the bus width. Values: 1 to 64
Gearing Ratio	Specify the gearing ratio. A 1x is an IO register. For gear ratio of 2x and 4x, the output width is 2 or 4 times the input width. The clock rate will be slower by the same amount. Values: 1x, 2x, 4x
Input, Output Clock Mode	Specifies the type of clock to use for the modes where no gearing is used. Values: Edge, Primary/Secondary
Data Delay	Specify the Data Delay type. By default, there is no delay in the data path. The delay may be set at configuration to values specified by CDEL and FDEL, set to match the primary or edge clock, or set to match the DLL Delay Controls (DCNTL bus). Values: None, DLL, Edge Clock, Primary Clock, ECLK Injection, User Defined
CDEL	Specify the coarse delay setting. Values: 0, 1, 2, 3
FDEL	Specify the fine delay setting. Values: 0 to 47
Use CLKDIV	Specify the use of CLKDIV element to get the slower clock. Using the clock and reset from the CLKDIV will reduce the skew between the two. This is useful for high-speed interfaces that have gearing.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 284: SDR Dialog Box for LatticeSC/M—Advanced Tab

Feature	Description
Use AIL	Specify that the Adaptive Input Logic (AIL) is used. This is available for configurations that contain an Input SDR. The AIL allows the I/O logic to automatically control coarse and fine delay settings to guarantee setup and hold times for a single I/O by sliding the data to a particular clock edge. The AIL features are not bus-based, but rather, per input port. Since the AIL will slide the data to a particular clock edge, the bits of a bus may not be latched on the same clock edge. This requires a training pattern to realign the individual bits of the input data bus to the same clock edge. For single bit instances, this training pattern is not required. An AIL deskew reference design is available from Lattice Semiconductor that provides an example of the circuitry and training pattern required to align the bits of a bus. Please contact Lattice Semiconductor for more information on this reference design.
AIL Acquisition Window (ps)	Specify the window size in ps where data transitions are not allowed. The AIL continuously delays data to keep the AIL Acquisition Window within the data pulse width away from the transition edges. Values: 400, 720, 1040, 1360
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195

Sin-Cos_Table

Look-up table for trigonometric sine and cosine values for an angle. The angle, Theta, is given in radians: 360° = 2π radians or 1 radian = 180°/π.

Table 285: Sin-Cos_Table Dialog Box

Feature	Description
Block Implementation	Specify if the module is implemented using EBR memory or LUTs. With Platform Manager, only LUT is available.
Input Theta Bit Width	Specify the width of the Theta input bus. This determines the precision with which the angle can be specified. The value of Theta will be in radians, from 0 to something less than 2π, depending on the width, and in the form of unsigned fixed-point data. The data will have a scaling factor of $\pi/2^{(width - 1)}$. For example, with an 8-bit Theta, π radians (180°) is given as $\pi \times 2^{(8 - 1)}/\pi = \pi \times 128/\pi$ = h'80.

Table 285: Sin-Cos_Table Dialog Box (Continued)

Feature	Description
Output Result Bit Width	Specify the width of the Sine and Cosine output buses.
Output Mode	Specify the outputs that are needed. More memory resources may be used when the Sin-Cos option is selected, but in some cases, this may be better than two independent Sin and Cos modules.
Use Tables for Quarter-wave Only	Select to reduce memory by only storing a quarter wave and deriving the rest of it with additional logic. If selected, see “Specify the number of pipeline stages” below to improve performance.
Use 1-bit for Signed Integer	Specify if the output values should be in twos complement format or sign-and-magnitude format. If this option is selected, the values will be in twos complement format with a scaling factor of $1/2^{(width - 1)}$. If this option is cleared, the values will be in sign-and-magnitude format with a scaling factor of $1/2^{(width - 2)}$.
Enable Input Registers	Specify if input registers are required.
Enable Output Registers	Specify if output registers are required. Output registers are automatically included in the LUT implementation.
Specify the number of pipeline stages	If Use Tables for Quarter-wave Only is selected (above), specify the number of register stages to insert to improve performance.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195

SLICE

Generic module used to generate custom configurations. Provides a variety of multiply/accumulate organizations and is implemented as embedded sysDSP blocks. The full SLICE configuration is recommended only if MULT, MAC, MULTADDSUB, and MULTADDSUBSUM configurations cannot satisfy the application requirement.

The module dialog box has two tabs:

- Configuration, below
- Register Setup, see Table 287 on page 1647

Slice includes optional input and output registers. There are also optional pipeline registers between the multipliers and the ALU.

In the Register Setup tab you can choose to enable or to bypass any of these registers. Normally you would leave them all enabled and instead use the Enable Input/Pipeline/Output Registers check boxes in the Configuration tab. Normally you want all registers of the same type set the same for the sake of timing.

In the Register Setup tab, you can also specify the clock, clock enable, and reset signal for each register.

Table 286: SLICE Dialog Box—Configuration Tab

Feature	Description
Mult and ALU Selection	**Enable MultA** – Enable the A multiplier and the AA and AB inputs. MultA must be disabled to do logical combinations of the ALU's B and C inputs (see "Operation" following), or to use some options for the ALU's A input (see "Input A (A Mux)" following). **Enable MultB** – Enable the B multiplier and the BA and BB inputs. MultB must be disabled to use some options for the ALU's B input (see "Input B (B Mux)" following). **Cascade Match Register** – Enable the Cascade Match Register of the B multiplier. **Select Shift Out B** – Enable the shift-out port B of the DSP block. **Enable ALU** – Enable the arithmetic logic unit that combines the results of multipliers A and B. Disable the ALU to get the results of the multipliers on separate output buses. **Operation** – Specify an arithmetic combination of the ALU's A, B, and C inputs; a logical combination of the B and C inputs; or dynamic selection of the operation. For the B and C logical combinations, clear **Enable MultA** (above). Inputs are defined below the Operation box. A DYNAMIC setting is available for all inputs. Dynamic mode settings for Opcode[3:0]: A + B + C: 0100 A - B + C: 0101 A + B - C: 0110 A - B - C: 0111 B XNOR C: 1100 B XOR C: 1110 B NAND C: 0000 B AND C: 1000 B OR C: 0011 B NOR C: 1011

Table 286: SLICE Dialog Box—Configuration Tab (Continued)

Feature	Description
	Input A (A Mux) – Specify the A input for the ALU. For the MultA option, select **Enable MultA** (above). Dynamic mode settings for Amuxsel[1:0]: ALU Feedback: 00 MultA: 01 A_ALU: 10 0: 11 **Input B (B Mux)** – Specify the B input for the ALU. For the MultB Shift 18 L and MultB options, select **Enable MultB** (above). Dynamic mode settings for Bmuxsel[1:0]: MultB Shift 18 L: 00 MultB: 01 B_ALU: 10 0: 11
	Input C (C Mux) – Specify the C input for the ALU. Dynamic mode settings for Cmuxsel[2:0]: 0: 000 CIN Shift 18 R: 001 CIN: 010 C_ALU: 011 A_ALU: 100 ALU Feedback: 101 RNDtoPN: 110 RNDtoPNM1: 111
	MCPAT_SOURCE – Specify whether the source of the MEM cell pattern is static or dynamic. If static, specify the pattern in the MCPAT box. If dynamic, send the pattern through the C bus. MCPAT_SOURCE and MASKPAT_SOURCE cannot both be dynamic. Not available when Input C is set to C_ALU or A_ALU. **MASKPAT_SOURCE** – Specify whether the source of the mask for EQPAT/EQPATB is static or dynamic. If static, specify the mask in the MASKPAT box. If dynamic, send the mask through the C bus. MCPAT_SOURCE and MASKPAT_SOURCE cannot both be dynamic. Available only when Input C is set to C_ALU or A_ALU. **MCPAT** – Specify the MEM cell pattern in hexadecimal. **MASKPAT** – Specify the mask for EQPAT/EQPATB in hexadecimal. **MASK01** – Specify the mask for EQZM/EQOM in hexadecimal. **RNDPAT** – Specify the rounding pattern in hexadecimal. **n for RNDPAT** – Available only when Input C is set to RNDtoPN, RNDtoPNM1, or DYNAMIC.
	Reset Mode – Specify whether the reset is synchronous or asynchronous.

Table 286: SLICE Dialog Box—Configuration Tab (Continued)

Feature	Description
Input Selection	**Width** – Specify the size of the inputs. C's width can be set only when Input C is set to C_ALU, A_ALU, or DYNAMIC. **Source** – Specify if the source of the input is Parallel for generic routing, Shift for the shift outputs of an adjacent DSP block, or Dynamic for dynamic selection of the source of the input. **Sign** – Specify the data format of the inputs as signed, unsigned, or dynamic selection of the sign.
	Enable Input Registers – Enable the input registers as specified in the Register Setup tab. **Enable Pipeline Registers** – Enable the pipeline registers as specified in the Register Setup tab. **Enable Output Registers** – Enable the output registers and the flag register as specified in the Register Setup tab.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 287: SLICE Dialog Box—Register Setup Tab

Feature	Description
Enable Input Register AA, AB, BA, BB, C Enable ALU Opcode Register	Input registers for the A and B multipliers, the C ALU, and the ALU Opcode signal.
Enable MultA Pipeline Register Enable MultB Pipeline Register Enable ALU Opcode Pipeline Register	Pipeline registers from the multiplier outputs to the ALU and from the ALU Opcode (or the Opcode register if enabled) to the ALU. If the ALU is disabled (by clearing Enable ALU in the Configuration tab), the multiplier pipeline registers go to the PA and PB outputs of the Slice module.
Enable MultA Output Register Enable MultB Output Register Enable ALU Output and Flag Register	Output registers for the multipliers and the ALU.
Bus Ordering Style	Specify if the inputs and output are big endian (MSB:LSB) or little endian (LSB:MSB), or none.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.

Table 287: SLICE Dialog Box—Register Setup Tab (Continued)

Feature	Description
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also
- "Generating a Module or IP with IPexpress" on page 195
- "How to Design with sysDSP" on page 1172
- TN1267, *ECP5 sysDSP Usage Guide*

pmi_dsp_preadd_slice

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_dsp_preadd_slice.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	M0A	Bus	(pmi_data_m0a_width-1):0
I	M0B	Bus	(pmi_data_m0b_width-1):0
I	M1A	Bus	(pmi_data_m1a_width-1):0
I	M1B	Bus	(pmi_data_m1b_width-1):0
I	M0C	Bus	(pmi_data_m0c_width-1):0
I	M1C	Bus	(pmi_data_m1c_width-1):0
I	SIGNM0A	Bit	N/A
I	SIGNM0B	Bit	N/A
I	SIGNM1A	Bit	N/A
I	SIGNM1B	Bit	N/A
I	SOURCEM0A	Bit	N/A
I	SOURCEM1A	Bit	N/A
I	CFB	Bus	53:0
I	CIN	Bus	53:0
I	SIGNCIN	Bit	N/A
I	SRIA	Bus	17:0
I	SRIB	Bus	17:0
I	AMUXSEL	Bus	1:0
I	BMUXSEL	Bus	1:0
I	CMUXSEL	Bus	2:0
I	OPPRE	Bus	1:0
I	OP	Bus	3:0
I	CLK	Bus	3:0
I	CE	Bus	3:0
I	RST	Bus	3:0
O	R	Bus	53:0
O	CO	Bus	53:0
O	SIGNR	Bit	N/A
O	SROA	Bus	17:0
O	SROA	Bus	17:0

Verilog pmi_dsp_preadd_slice Definition

```
module pmi_dsp_preadd_slice #(
    parameter pmi_data_m0a_width = 18,
    parameter pmi_data_m0b_width = 18,
    parameter pmi_data_m1a_width = 18,
    parameter pmi_data_m1b_width = 18,
    parameter pmi_data_m0c_width = 18,
    parameter pmi_data_m1c_width = 18,
    parameter pmi_gsr = "enable",
    parameter pmi_family = "ECP5U",
    parameter pmi_reg_input_m0a_pre_clk = "CLK0",
    parameter pmi_reg_input_m0a_pre_ce = "CE0",
    parameter pmi_reg_input_m0a_pre_rst = "RST0",
    parameter pmi_reg_input_m0b_pre_clk = "CLK0",
    parameter pmi_reg_input_m0b_pre_ce = "CE0",
    parameter pmi_reg_input_m0b_pre_rst = "RST0",
    parameter pmi_reg_input_m0a_mult_clk = "CLK0",
    parameter pmi_reg_input_m0a_mult_ce = "CE0",
    parameter pmi_reg_input_m0a_mult_rst = "RST0",
    parameter pmi_reg_input_m0b_mult_clk = "CLK0",
    parameter pmi_reg_input_m0b_mult_ce = "CE0",
    parameter pmi_reg_input_m0b_mult_rst = "RST0",
    parameter pmi_reg_input_m1a_pre_clk = "CLK0",
    parameter pmi_reg_input_m1a_pre_ce = "CE0",
    parameter pmi_reg_input_m1a_pre_rst = "RST0",
    parameter pmi_reg_input_m1b_pre_clk = "CLK0",
    parameter pmi_reg_input_m1b_pre_ce = "CE0",
    parameter pmi_reg_input_m1b_pre_rst = "RST0",
    parameter pmi_reg_input_m1a_mult_clk = "CLK0",
    parameter pmi_reg_input_m1a_mult_ce = "CE0",
    parameter pmi_reg_input_m1a_mult_rst = "RST0",
    parameter pmi_reg_input_m1b_mult_clk = "CLK0",
    parameter pmi_reg_input_m1b_mult_ce = "CE0",
    parameter pmi_reg_input_m1b_mult_rst = "RST0",
    parameter pmi_reg_input_m0c_clk = "CLK0",
    parameter pmi_reg_input_m0c_ce = "CE0",
    parameter pmi_reg_input_m0c_rst = "RST0",
    parameter pmi_reg_input_m1c_clk = "CLK0",
    parameter pmi_reg_input_m1c_ce = "CE0",
    parameter pmi_reg_input_m1c_rst = "RST0",
    parameter pmi_reg_input_cfb_clk = "CLK0",
    parameter pmi_reg_input_cfb_ce = "CE0",
    parameter pmi_reg_input_cfb_rst = "RST0",
    parameter pmi_reg_pipeline0_clk = "CLK0",
    parameter pmi_reg_pipeline0_ce = "CE0",
    parameter pmi_reg_pipeline0_rst = "RST0",
    parameter pmi_reg_pipeline1_clk = "CLK0",
    parameter pmi_reg_pipeline1_ce = "CE0",
    parameter pmi_reg_pipeline1_rst = "RST0",
    parameter pmi_reg_output_clk = "CLK0",
    parameter pmi_reg_output_ce = "CE0",
    parameter pmi_reg_output_rst = "RST0",
    parameter pmi_reg_oppre0_clk = "CLK0",
    parameter pmi_reg_oppre0_ce = "CE0",
    parameter pmi_reg_oppre0_rst = "RST0",
    parameter pmi_reg_oppre1_clk = "CLK0",
    parameter pmi_reg_oppre1_ce = "CE0",
    parameter pmi_reg_oppre1_rst = "RST0",
    parameter pmi_reg_opcodein0_clk = "CLK0",
    parameter pmi_reg_opcodein0_ce = "CE0",
```

```
    parameter pmi_reg_opcodein0_rst = "RST0",
    parameter pmi_reg_opcodein1_clk = "CLK0",
    parameter pmi_reg_opcodein1_ce = "CE0",
    parameter pmi_reg_opcodein1_rst = "RST0",
    parameter pmi_cascade_match_reg = "off",
    parameter pmi_m0_sourceb_mode = "B",
    parameter pmi_m1_sourceb_mode = "B",
    parameter pmi_m0_sourcea_mode = "OPA0",
    parameter pmi_m1_sourcea_mode = "OPA1",
    parameter pmi_pre0_sourcea_mode = "A_SHIFT",
    parameter pmi_pre0_sourceb_mode = "SHIFT",
    parameter pmi_pre0_symmetry_mode = "DIRECT",
    parameter pmi_pre1_sourcea_mode = "A_SHIFT",
    parameter pmi_pre1_sourceb_mode = "SHIFT",
    parameter pmi_pre1_fb_mux = "SHIFT",
    parameter pmi_pre1_symmetry_mode = "DIRECT",
    parameter pmi_highspeed_clk = "NONE",
    parameter pmi_clk0_div = "ENABLED",
    parameter pmi_clk1_div = "ENABLED",
    parameter pmi_clk2_div = "ENABLED",
    parameter pmi_clk3_div = "ENABLED",
    parameter module_type = "pmi_dsp_preadd_slice")

  (input [pmi_data_m0a_width-1:0] M0A,
   input [pmi_data_m0b_width-1:0] M0B,
   input [pmi_data_m1a_width-1:0] M1A,
   input [pmi_data_m1b_width-1:0] M1B,
   input [pmi_data_m0c_width-1:0] M0C,
   input [pmi_data_m1c_width-1:0] M1C,
   input SIGNM0A, SIGNM0B, SIGNM1A, SIGNM1B,
   input SOURCEM0A, SOURCEM1A,
   input [53:0] CFB, CIN,
   input SIGNCIN,
   input [17:0] SRIA, SRIB,
   input [1:0] AMUXSEL, BMUXSEL,
   input [2:0] CMUXSEL,
   input [1:0] OPPRE,
   input [3:0] OP,
   input [3:0] CLK, CE, RST,
   output [53:0] R, CO,
   output SIGNR,
   output [17:0] SROA, SROB)/*synthesis syn_black_box*/;
```

VHDL pmi_dsp_preadd_slice Definition

```
component pmi_dsp_preadd_slice is
    generic (
       pmi_data_m0a_width : integer := 18;
       pmi_data_m0b_width : integer := 18;
       pmi_data_m1a_width : integer := 18;
       pmi_data_m1b_width : integer := 18;
       pmi_data_m0c_width : integer := 18;
       pmi_data_m1c_width : integer := 18;
       pmi_gsr : string := "enable";
       pmi_family : string := "ECP5U";
       pmi_reg_input_m0a_pre_clk : string := "CLK0";
       pmi_reg_input_m0a_pre_ce : string := "CE0";
       pmi_reg_input_m0a_pre_rst : string := "RST0";
       pmi_reg_input_m0b_pre_clk : string := "CLK0";
```

```
pmi_reg_input_m0b_pre_ce : string := "CE0";
pmi_reg_input_m0b_pre_rst : string := "RST0";
pmi_reg_input_m0a_mult_clk : string := "CLK0";
pmi_reg_input_m0a_mult_ce : string := "CE0";
pmi_reg_input_m0a_mult_rst : string := "RST0";
pmi_reg_input_m0b_mult_clk : string := "CLK0";
pmi_reg_input_m0b_mult_ce : string := "CE0";
pmi_reg_input_m0b_mult_rst : string := "RST0";
pmi_reg_input_m1a_pre_clk : string := "CLK0";
pmi_reg_input_m1a_pre_ce : string := "CE0";
pmi_reg_input_m1a_pre_rst : string := "RST0";
pmi_reg_input_m1b_pre_clk : string := "CLK0";
pmi_reg_input_m1b_pre_ce : string := "CE0";
pmi_reg_input_m1b_pre_rst : string := "RST0";
pmi_reg_input_m1a_mult_clk : string := "CLK0";
pmi_reg_input_m1a_mult_ce : string := "CE0";
pmi_reg_input_m1a_mult_rst : string := "RST0";
pmi_reg_input_m1b_mult_clk : string := "CLK0";
pmi_reg_input_m1b_mult_ce : string := "CE0";
pmi_reg_input_m1b_mult_rst : string := "RST0";
pmi_reg_input_m0c_clk : string := "CLK0";
pmi_reg_input_m0c_ce : string := "CE0";
pmi_reg_input_m0c_rst : string := "RST0";
pmi_reg_input_m1c_clk : string := "CLK0";
pmi_reg_input_m1c_ce : string := "CE0";
pmi_reg_input_m1c_rst : string := "RST0";
pmi_reg_input_cfb_clk : string := "CLK0";
pmi_reg_input_cfb_ce : string := "CE0";
pmi_reg_input_cfb_rst : string := "RST0";
pmi_reg_pipeline0_clk : string := "CLK0";
pmi_reg_pipeline0_ce : string := "CE0";
pmi_reg_pipeline0_rst : string := "RST0";
pmi_reg_pipeline1_clk : string := "CLK0";
pmi_reg_pipeline1_ce : string := "CE0";
pmi_reg_pipeline1_rst : string := "RST0";
pmi_reg_output_clk : string := "CLK0";
pmi_reg_output_ce : string := "CE0";
pmi_reg_output_rst : string := "RST0";
pmi_reg_oppre0_clk : string := "CLK0";
pmi_reg_oppre0_ce : string := "CE0";
pmi_reg_oppre0_rst : string := "RST0";
pmi_reg_oppre1_clk : string := "CLK0";
pmi_reg_oppre1_ce : string := "CE0";
pmi_reg_oppre1_rst : string := "RST0";
pmi_reg_opcodein0_clk : string := "CLK0";
pmi_reg_opcodein0_ce : string := "CE0";
pmi_reg_opcodein0_rst : string := "RST0";
pmi_reg_opcodein1_clk : string := "CLK0";
pmi_reg_opcodein1_ce : string := "CE0";
pmi_reg_opcodein1_rst : string := "RST0";
pmi_cascade_match_reg : string := "off";
pmi_m0_sourceb_mode : string := "B";
pmi_m1_sourceb_mode : string := "B";
pmi_m0_sourcea_mode : string := "OPA0";
pmi_m1_sourcea_mode : string := "OPA1";
pmi_pre0_sourcea_mode : string := "A_SHIFT";
pmi_pre0_sourceb_mode : string := "SHIFT";
pmi_pre0_symmetry_mode : string := "DIRECT";
pmi_pre1_sourcea_mode : string := "A_SHIFT";
```

```
        pmi_pre1_sourceb_mode : string := "SHIFT";
        pmi_pre1_fb_mux : string := "SHIFT";
        pmi_pre1_symmetry_mode : string := "DIRECT";
        pmi_highspeed_clk : string := "NONE";
        pmi_clk0_div : string := "ENABLED";
        pmi_clk1_div : string := "ENABLED";
        pmi_clk2_div : string := "ENABLED";
        pmi_clk3_div : string := "ENABLED";
        module_type : string := "pmi_dsp_preadd_slice"
    );
    port (
        M0A : in std_logic_vector((pmi_data_m0a_width-1) downto
0);
        M0B : in std_logic_vector((pmi_data_m0b_width-1) downto
0);
        M1A : in std_logic_vector((pmi_data_m1a_width-1) downto
0);
        M1B : in std_logic_vector((pmi_data_m1b_width-1) downto
0);
        M0C : in std_logic_vector((pmi_data_m0c_width-1) downto
0);
        M1C : in std_logic_vector((pmi_data_m1c_width-1) downto
0);
        SIGNM0A, SIGNM0B, SIGNM1A, SIGNM1B : in std_logic;
        SOURCEM0A, SOURCEM1A : in std_logic;
        CFB, CIN : in std_logic_vector(53 downto 0);
        SIGNCIN : in std_logic;
        SRIA, SRIB : in std_logic_vector(17 downto 0);
        AMUXSEL, BMUXSEL : in std_logic_vector(1 downto 0);
        CMUXSEL : in std_logic_vector(2 downto 0);
        OPPRE : in std_logic_vector(1 downto 0);
        OP : in std_logic_vector(3 downto 0);
        CLK, CE, RST : in std_logic_vector(3 downto 0);
        R, CO : out std_logic_vector(53 downto 0);
        SIGNR : out std_logic;
        SROA, SROB : out std_logic_vector(17 downto 0)
    );
  end component pmi_dsp_preadd_slice;
```

Parameter Names and Values

The following table contains the pmi_dsp_preadd_slice module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value
pmi_data_m0a_width	2 to 18	18
pmi_data_m0b_width	2 to 18	18
pmi_data_m1a_width	2 to 18	18
pmi_data_m1b_width	2 to 18	18
pmi_data_m0c_width	2 to 18	18
pmi_data_m1c_width	2 to 18	18
pmi_gsr	“enable”	“disable”	“enable”

Parameter Name	Acceptable Values	Default Value
pmi_family	“ECP5U”	“ECP5U”
pmi_reg_input_m0a_pre_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m0a_pre_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m0a_pre_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m0b_pre_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m0b_pre_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m0b_pre_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m0a_mult_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m0a_mult_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m0a_mult_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m0b_mult_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m0b_mult_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m0b_mult_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m1a_pre_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m1a_pre_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m1a_pre_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m1b_pre_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m1b_pre_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m1b_pre_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m1a_mult_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m1a_mult_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m1a_mult_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m1b_mult_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m1b_mult_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m1b_mult_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m0c_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m0c_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m0c_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_m1c_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_m1c_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_input_m1c_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_input_cfb_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_input_cfb_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”

Parameter Name	Acceptable Values	Default Value
pmi_reg_input_cfb_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_pipeline0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_pipeline0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_pipeline0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_pipeline1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_pipeline1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_pipeline1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_output_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_output_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_output_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_oppre0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_oppre0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_oppre0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_oppre1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_oppre1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_oppre1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_opcodein0_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_opcodein0_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_opcodein0_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_reg_opcodein1_clk	“CLK0” | “CLK1” | “CLK2” | “CLK3”	“CLK0”
pmi_reg_opcodein1_ce	“CE0” | “CE1” | “CE2” | “CE3”	“CE0”
pmi_reg_opcodein1_rst	“RST0” | “RST1” | “RST2” | “RST3”	“RST0”
pmi_cascade_match_reg	“on” | “off”	“off”
pmi_m0_sourceb_mode	“B” | “C” | “HIGHSPEED”	“B”
pmi_m1_sourceb_mode	“B” | “C” | “HIGHSPEED”	“B”
pmi_m0_sourcea_mode	“OPA0” | “A” | “A_SHIFT”	“OPA0”
pmi_m1_sourcea_mode	“OPA1” | “A” | “A_SHIFT”	“OPA1”
pmi_pre0_sourcea_mode	“A_SHIFT” | “C_SHIFT” | “A_C_DYNAMIC” | “HIGHSPEED”	“A_SHIFT”
pmi_pre0_sourceb_mode	“SHIFT” | “PARALLEL” | “INTERNAL”	“SHIFT”
pmi_pre0_symmetry_mode	“DIRECT” | “INTERNAL”	“DIRECT”
pmi_pre1_sourcea_mode	“A_SHIFT” | “C_SHIFT” | “A_C_DYNAMIC” | “HIGHSPEED”	“A_SHIFT”
pmi_pre1_sourceb_mode	“SHIFT” | “PARALLEL” | “INTERNAL”	“SHIFT”
pmi_pre1_fb_mux	“SHIFT” | “SHIFT_BYPASS” | “DISABLED”	“SHIFT”

Parameter Name	Acceptable Values	Default Value
pmi_pre1_symmetry_mode	“DIRECT”	“INTERNAL”	“DIRECT”
pmi_highspeed_clk	“CLK0”	“CLK1”	“CLK2”	“CLK3”	“CLK0”
pmi_clk0_div	“ENABLED”	“DISABLED”	“ENABLED”
pmi_clk1_div	“ENABLED”	“DISABLED”	“ENABLED”
pmi_clk2_div	“ENABLED”	“DISABLED”	“ENABLED”
pmi_clk2_div	“ENABLED”	“DISABLED”	“ENABLED”

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

Subtractor

A two-input subtractor that performs signed or unsigned subtraction of the data from input ports DataA and DataB. If the optional port Cin is used, the subtraction result DataA - DataB + Cin is returned to the output port Result. The optional Cout port represents carry-out and the optional Overflow port detects overflow conditions when operating on signed data.

Table 288: Subtractor Dialog Box

Feature	Description
Specify the Data Width of the Subtractor	Specifies the input bus width of the module.
Specify the Representation of the Subtractor	Selections can be either signed or unsigned. Choosing the signed representation will keep the signed bit on the output. Unsigned will remove the signed bit from the output.
Complex Inputs	Allows implementation of complex subtractor.
Use Carry-in port	Adds a carry-in port to the module. This port will be Active Low.
Specify the Carry-out Port	Uses the carry-out port, overflow port, or none at all. Overflow port is allowed only when the representation is signed.
Enable Output Register	Allows the implementation of registers on the outputs.
Specify number of pipeline stages	Specifies the number of register stages inserted into the subtractor. More stages improves achievable frequency of operation, but the latency for the output increases.
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.

Table 288: Subtractor Dialog Box (Continued)

Feature	Description
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 289: Subtractor Ports

Name	Description
Cin (optional)	Carry-in to the low-order bit.
DataA[size-1..0]	Minuend
DataB[size-1..0]	Subtrahend
Result[size-1..0]	DataA[size-1..0] - DataB[size-1..0] + Cin - 1
Cout (optional)	Carry-out of the MSB to detect overflow in unsigned representation. When Cout is prepended to the Result, the result is a vector that always has sufficient precision to represent the result of the operation, {Cout, Result} = DataA + DataB + Cin. Appears as a port of the module when representation is set to unsigned and appears as an internal signal of the module when representation is set to signed. The signal is active low.
Overflow (optional)	Result exceeds available precision when used with signed representation. XOR of the carry into the MSB and Cout. The Overflow port only produces meaningful output if representation is set to signed.

See Also

- “Use PMI or IPexpress?” on page 191
- “Generating a Module or IP with IPexpress” on page 195

pmi_sub

The following are port descriptions, Verilog and VHDL definitions, and parameter descriptions for pmi_sub.

Port Description

Input/Output	Port Name	Type	Size (Buses Only)
I	DataA	Bus	(pmi_data_width - 1):0
I	DataB	Bus	(pmi_data_width - 1):0
I	Cin	Bit	N/A
O	Result	Bus	(pmi_data_width - 1):0
O	Cout	Bit	N/A
O	Overflow	Bit	N/A

Verilog pmi_sub Definition

```
module pmi_sub
  #(parameter pmi_data_width = 8,
    parameter pmi_result_width = 8,
    parameter pmi_sign = "off",
    parameter pmi_family = "EC",
    parameter module_type = "pmi_sub"
    )

    (
     input [pmi_data_width-1:0] DataA,
     input [pmi_data_width-1:0] DataB,
     input Cin,
     output [pmi_data_width-1:0] Result,
     output Cout,
     output Overflow)/*synthesis syn_black_box */;
endmodule // pmi_sub
```

VHDL pmi_sub Definition

```
component pmi_sub is
     generic (
       pmi_data_width : integer := 8;
       pmi_result_width : integer := 8;
       pmi_sign : string := "off";
       pmi_family : string := "EC";
       module_type : string := "pmi_sub"

    );
    port (
     DataA : in std_logic_vector(pmi_data_width-1 downto 0);
     DataB : in std_logic_vector(pmi_data_width-1 downto 0);
     Cin: in std_logic;
     Result : out std_logic_vector(pmi_data_width-1 downto 0);
     Cout: out std_logic;
     Overflow: out std_logic
   );
  end component pmi_sub;
```

Parameter Names and Values

The following table contains the pmi_sub module parameters, acceptable values, and default value.

Parameter Name	Acceptable Values	Default Value	Notes
pmi_data_width	1 to 64	8	The LatticeSC/M device families data and result widths range from 1 to 256.
pmi_result_width	1 to 64	8	The LatticeSC/M device families data and result widths range from 1 to 256.
pmi_sign	“on” | “off”	“off”	
pmi_family	“EC” | “ECP” | “ECP2” | “ECP2M” | “ECP3” | “ECP5U” | “ECP5UM” | “SC” | “SCM” | “XP” | “XP2” | “XO” | “XO2” | “XO3L” | “LPTM”	“EC”	

See Also

- “Use PMI or IPexpress?” on page 191
- “Using PMI” on page 200

System_Bus

A LatticeSC/M system bus.

In addition to the usual files, the System Bus module includes a <*module_name*>.txt file. This file is used by simulation and bitgen to configure the modules. When you import the source files for this module into the design project, also import the <*module_name*>.txt file as a source. It will show up under Documents in the Files tab.

The module dialog box has seven tabs:

- MPI, below
- User Master, see Table 291 on page 1660
- User Slave, see Table 292 on page 1660
- SMI, Table 293 on page 1661
- PCS, see Table 294 on page 1661
- Global, see Table 295 on page 1662
- Advanced, see Table 296 on page 1662

Table 290: System_Bus Dialog Box—MPI Tab

Feature	Description
Enable MPI	Provides pins for the MPI interface on the system bus module. Selecting this option generates top-level MPI ports. Any tristate output ports and bidirectional buffers are also instantiated in the netlist. MPI is not available with the smallest LatticeSC/M packages.
Bus Width	Sets the bus width of the MPI data/parity buses. The width of the MPI data bus can be 8, 16, or 32. This determines the number of I/O pads locked for the MPI.
Parity Enable	Enables the creation of the parity pins on the MPI bus. When selected, the parity bits are connected to the top-level ports. The number of parity bits is determined by the MPI data bus width.
DMA Interface	Provides ports for the DFA interface on the system bus module.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 291: System_Bus Dialog Box—User Master Tab

Feature	Description
Enable User Master Interface	Provides ports for the user master interface (UMI) on the system bus module.
User Master Sync. to Systembus Clock (HCLK)	The system bus uses two modes of clocking: synchronous and asynchronous. This option will select the UMI to be a synchronous transaction. The asynchronous interface uses a small internal FIFO to cross clock domains from the UMI to the system bus.
Automatic Retry	Enables the UMI to automatically retry a transaction if necessary.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 292: System_Bus Dialog Box—User Slave Tab

Feature	Description
Enable User Slave Interface	Provides ports for the user slave interface (USI) on the system bus module.
User Slave Sync. to Systembus Clock (HCLK)	The system bus uses two modes of clocking: synchronous and asynchronous. This option will select the USI to be a synchronous transaction. The asynchronous interface uses a small internal FIFO to cross clock domains from the USI to the system bus.

Table 292: System_Bus Dialog Box—User Slave Tab (Continued)

Feature	Description
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 293: System_Bus Dialog Box—SMI Tab

Feature	Description
Enable Serial Management Interface	Provides ports for the user slave interface (USI) on the system bus module.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 294: System_Bus Dialog Box—PCS Tab

Feature	Description
Enable PCS Control/Status Ports	Include PCS Interface pins on the System Bus clock interface for the selected PCS quads. Select those PCS Quads which require System Bus control and/or monitoring.
Select Multi-Quad Alignment	Select **Enable Multi-Quad Alignment**. Available if “Enable PCS Control/Status Ports” is selected for two or more PCS quads. Then choose one of four Multi-Quad Alignment input clocks (0, 1, 2, or 3) for each quad that requires multi-quad alignment. All quads to be aligned to each other must use the same Multi-Quad Alignment input clock. Select “-” for any PCS quad that does not require Multi-Quad Alignment.
Enable Multi-Chip Alignment	Enable PCS Control/Status Ports has been selected for at least one PCS Quad. All PCS Quads which are to be multi-chip aligned must be set to the Group 0 clock.
Local Device	If Enable Multi-Chip Alignment is selected, specify if the local device is a slave or the master.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 295: System_Bus Dialog Box—Global Tab

Feature	Description
Parity	This selection controls the parity that is generated and checked by the system bus.
MPI Priority	The system bus can be configured as a multi-master bus. This selection sets the priority of the MPI. To select this option, first select Enable MPI in the MPI tab and Enable user Master Interface in the User_Master tab.
User Master Priority	The system bus can be configured as a multi-master bus. This selection sets the priority of the UMI. To select this option, first select Enable MPI in the MPI tab and Enable user Master Interface in the User_Master tab.
Systembus Reset by MPI	Allows the user to reset the system bus through a run time register only accessible from the MPI. To select this option, first select **Enable MPI** in the MPI tab.
Systembus Reset by User Master	Allows the user to reset the system bus through a run time register only accessible from the UMI. To select this option, first select **Enable User Master Interface** in the User_Master tab.
Systembus Clock	Selects the source of the internal system bus clock.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

Table 296: System_Bus Dialog Box—Advanced Tab

Feature	Description
Read Only Word #1-6	Read only words that are set in the FPGA bitstream. These values are written to specific read-only registers in the system bus for run time access.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195
- TN1085, *LatticeSC MPI/System Bus*

Tag Memory

A one-page FLASH non-volatile memory accessible by the hard-wired Serial Peripheral Interface port or the JTAG port. This stand-alone TAG memory is ideal for scratch pad memory for mission critical data, board serialization, board revision log, and programmed pattern identification.

Table 297: Tag Memory Dialog Box

Feature	Description
Tag Memory Size	Displays the available memory, in bytes, for the selected device. This number cannot be changed by the user.
Memory File	Allows you to select a memory initialization (.mem) file for the module. See "Creating a Memory Initialization File with Memory Generator" on page 203.
Memory File Format	Allows you to choose Binary, Hex, or Addressed Hex format.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- "Generating a Module or IP with IPexpress" on page 195
- DS1009 - LatticeXP2 Family Data Sheet
- TN1137, *LatticeXP2 Memory Usage Guide*
- TN1141, *LatticeXP2 sysCONFIG Usage Guide*

WIDE_MUX

Multiplexes several input buses onto a single output bus.

Table 298: WIDE_MUX Dialog Box

Feature	Description
Width	Specifies the width of the input data buses and the Result bus.
Number of Inputs	Specifies the number of input data buses. The size of the Sel (select) bus adjusts automatically.
Reset Mode	Specify either a synchronous or asynchronous reset.
Enable Fully Pipelined Mode	Enables the output registers of all stages.
Enable Input Register	Enables the registers on the inputs.
Enable Output Register	Enables the registers on the output.

Table 298: WIDE_MUX Dialog Box (Continued)

Feature	Description
Bus Ordering Style	Specifies whether the numbers should begin at the high end (“Big Endian”- BusA[7 to 0]) or at the low end (“Little Endian”-BusA[0 to 7]). Selecting None prevents any bus notation and writes out all individual pins.
Import IPX to Diamond project	Adds the customized module to the currently open project. Not available if running IPexpress as a stand-alone tool. Not available in Clarity Designer.
Generate	Builds the module as specified.
Close	Closes the dialog box without generating the module.
Help	Opens the online Help.

See Also

- “Generating a Module or IP with IPexpress” on page 195

Chapter 19

FPGA Libraries Reference Guide

Lattice supports some libraries used in designing FPGAs with different device architectures in a number of CAE synthesis, schematic capture, and simulation platforms. These libraries are the main front-end design libraries for Lattice FPGAs. Logic design primitives in these libraries offer flexibility and efficiency to facilitate building specific applications with Lattice devices. Many primitives can be used in multiple Lattice device architectures. With Schematic Editor, you can place the primitive symbols from the Lattice Symbol Library, lattice.lib, which is composed of primitives compatible with most Lattice FPGA device families. For macro-sized primitives like architectural blocks, arithmetic, or memories, use IPexpress™ to configure and generate schematic symbols and files for implementation. You can also use physical macros that you create in EPIC. See the appropriate topic in the online help system for more information.

A specific primitive can be found according to the device family and functional category. Primitives available to each of the following device families are listed according to appropriate functional categories.

- “Primitive Library - ECP5” on page 1686
- “Primitive Library - LatticeECP/EC and LatticeXP” on page 1695
- “Primitive Library - LatticeECP2/M” on page 1707
- “Primitive Library - LatticeECP3” on page 1717
- “Primitive Library - LatticeSC/M” on page 1727
- “Primitive Library - LatticeXP2” on page 1739
- “Primitive Library - LIFMD” on page 1759
- “Primitive Library - MachXO and Platform Manager” on page 1780
- “Primitive Library - MachXO2 and Platform Manager 2” on page 1788
- “Primitive Library - MachXO3D” on page 1799
- “Primitive Library - MachXO3L” on page 1809
- “Primitive Library - LFMNX” on page 1749

The “Alphanumeric Primitives List” section contains descriptions of all available primitives in their alphanumeric order. The following information is provided for each primitive, where applicable:

Table 299: Information Provided for Each Primitive

Fields	Description
Name	Primitive name
Definition	Brief description of primitive
Architectures Supported	Index of FPGA families supported by the primitive
Port Interface Symbol	Graphic to represent the primitive port interface. Data, address, clock, clock enable type ports appear on the left-hand side of the block. Synchronous control ports appear on the top of the symbol, asynchronous control ports on the bottom, and output ports on the right-hand side of the block. Some of the graphic symbols are shown in bus notation for proper layout. Those primitives must be instantiated in expanded bus notation format with each individual bit.
Attributes	Index of attributes compatible with the primitive. The first value is usually the default value, if it is not explicitly indicated. Attribute function, range, and port-to-attribute or attribute-to-attribute relationships for the primitive are noted. For descriptions of all the attributes used in primitives, see “List of Primitive-Specific HDL Attributes” on page 2445.
Description	Detailed description of primitive function. Exceptions are identified by device family. This section sometimes includes truth table, waveform, state diagram, or other graphical methods to illustrate behavior. References to appropriate Lattice technical notes are listed.
Port Description	Index of port names, polarity, and function.
Connectivity Rules	Some primitive ports, such as DDR and DSP blocks, are connected to dedicated routing with source or loads related to other primitives. This section describes connection rules.

Naming Conventions

The table shows the convention for naming all of the sequential primitives. Each primitive is identified using up to seven characters.

Flip-Flop/Latch Naming Conventions (name = abcdef)

Table 300: Naming Conventions

a=	F - Static implementation
b=	D - D type flip-flop
	J - J/K type flip-flop
	L - Cells contain a positive select front end (loadable)
	N - Cells contain a negative select front end (loadable)
	S - R-S type flip-flop
	T - Toggle type flip-flop
c=	Value - Number of clocks
d=	This parameter identifies the enable capability.
	S - No enable input
	P - Positive-level enable
	N - Negative-level enable
e=	This parameter identifies the clock capability
	1 - Positive-level sense (latch)
	2 - Negative edge-triggered (flip-flop)
	3 - Positive edge-triggered (flip-flop)
	4 - Negative-level sense (latch)
f=	A - No clear or preset inputs
	B - Positive-level asynchronous preset
	D - Positive-level asynchronous clear
	I - Positive-level synchronous clear
	J - Positive-level synchronous preset
	X - Standard primitive where GSR asynchronously clears or presets the flip-flop depending upon the function of the local clear or preset. If no local clear or preset is present (f=A), then GSR clears the register element.
	Y - Primitive is preset using GSR rather than cleared
	Z - Primitive not compatible with similar primitives available in the standard cell library

Example:

FD1P3BX is a single clock, positive edge-triggered, static, D-type flip-flop with a positive-level enable and a positive-level asynchronous preset.

Table 301: Example: FD1P3BX

FD1P3BX	a = F	Static cell
	b = D	D type
	c = 1	Single clock
	d = P	Positive-level enable
	e = 3	Positive edge-triggered
	f = B	Positive-level asynchronous preset

When creating designs from schematic or synthesis, please observe the following rules:

- Component, net, site, or instance names should be unique and independent of case. For example, if a net in a design is named "d0", then you cannot have an instance of AND2 named "D0".
- Component, net, site, or instance names cannot be any cell name.
- Component, net, site, or instance names cannot be GND, PWR, VSS, VDD, GSR, GSRNET, TSALL, TSALLNET, and so forth.
- Component, net, site, or instance names cannot contain preference keywords such as DIN, DOUT, SITE, COMP, and so forth.
- Component, net, site, or instance names cannot contain names starting with 0–9, /, \, +, –, and other special characters.
- Component, net, site, or instance names cannot be named using VHDL or Verilog keywords such as IN, OUT, INOUT, and so forth.

Note the following about numbering used throughout this Help:

- Least significant bits (LSBs) on a primitive are always determined by the lowest integer value (usually zero) expressed in a pin name input or output in a pin grouping just as most significant bits (MSBs) are always determined by the highest integer value.

For example, out of the pin group containing pins A0, A1, A2, and A3, A0 is the LSB and A3 is the MSB. The LSB or MSB is not affected by the order in which pins are numbered (that is, whether or not they are in ascending or descending order).

Memory Primitives Overview

The architectures of various Lattice FPGAs provide resources for on-chip memory intensive applications. The sysMEM™ embedded block RAM (EBR) complements the distributed PFU-based memory. Single-port RAM, dual-port

RAM, FIFO, and ROM memories can be constructed using the EBR. LUTs and PFU can implement distributed single-port RAM, dual-port RAM, and ROM. The Lattice Diamond software enables you to integrate the EBR- and PFU-based memories in various device families.

The EBR-based and PFU-based memory primitives are listed in this document. Designers can utilize the memory primitives in several ways via the IPexpress tool in the Diamond software. IPexpress allows you to specify the attributes of the memory application, such as required type and size. IPexpress takes the customized specification and constructs a netlist to implement the desired memory, using one or more of the memory primitives.

The available memory primitives include:

- "RAM_DP (Dual Port RAM)" on page 1669
- "RAM_DP_BE (Dual Port RAM with Byte Enable)" on page 1671
- "RAM_DP_TRUE (True Dual Port RAM)" on page 1672
- "RAM_DP_TRUE_BE (True Dual Port RAM with Byte Enable)" on page 1674
- "RAM_DQ (Single Port RAM)" on page 1675
- "RAM_DQ_BE (Single Port RAM with Byte Enable)" on page 1676
- "ROM (Read Only Memory)" on page 1678
- "Distributed_DPRAM (Distributed Dual Port RAM)" on page 1679
- "Distributed_ROM (Distributed Read Only Memory)" on page 1680
- "Distributed_SPRAM (Distributed Single Port RAM)" on page 1680
- "FIFO (First In First Out Single Clock)" on page 1681
- "FIFO_DC (First In First Out Dual Clock)" on page 1682
- "Shift Registers (Distributed RAM Shift Register)" on page 1684

RAM_DP (Dual Port RAM)

Implementation: EBR

Table 302: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_wr_addr_width – 1) : 0
I	RdAddress	Bus	(pmi_rd_addr_width – 1) : 0
I	Data	Bus	(pmi_wr_data_width – 1) : 0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A

Table 302: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_rd_data_width – 1) : 0

Table 303: Parameters

Name	Value	Default
pmi_rd_addr_depth [1]	2 to 65536	512
pmi_rd_addr_width [1]	1 to 16	9
pmi_rd_data_width	1 to 256	18
pmi_wr_addr_depth [1]	2 to 65536	512
pmi_wr_addr_width [1]	1 to 16	9
pmi_wr_data_width	1 to 256	18
pmi_regmode	"reg" | "noreg"	"reg"
pmi_gsr	"enable" | "disable"	"disable"
pmi_resetmode [2]	"async" | "sync"	"sync"
pmi_optimization	"area" | "speed"	"speed"
pmi_init_file	*<string>*	"none"
pmi_init_file_format	"binary" | "hex"	"binary"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Note

1. For SC, SCM, ECP2, ECP2M, XP2, and ECP3 FPGA device families, the read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
2. For ECP3, "async" is not a valid option.

RAM_DP_BE (Dual Port RAM with Byte Enable)

Implementation: EBR

Table 304: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_wr_addr_width – 1) : 0
I	RdAddress	Bus	(pmi_rd_addr_width – 1) : 0
I	Data	Bus	(pmi_wr_data_width – 1) : 0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_rd_data_width – 1) : 0
I	ByteEn	Bus	((pmi_wr_data_width + pmi_byte_size – 1) / pmi_byte_size – 1) : 0

Table 305: Parameters

Name	Value	Default
pmi_rd_addr_depth[1]	2 to 65536	512
pmi_rd_addr_width[1]	1 to 16	9
pmi_rd_data_width	1 to 256	18
pmi_wr_addr_depth[1]	2 to 65536	512
pmi_wr_addr_width[1]	1 to 16	9
pmi_wr_data_width	1 to 256	18
pmi_regmode	"reg" | "noreg"	"reg"
pmi_gsr	"enable" | "disable"	"disable"
pmi_resetmode[2]	"async" | "sync"	"sync"
pmi_optimization	"area" | "speed"	"speed"
pmi_init_file	*<string>*	"none"
pmi_init_file_format	"binary" | "hex"	"binary"

Table 305: Parameters (Continued)

Name	Value	Default
pmi_family	"XP2" | "SC" | "SCM" | "ECP2" | "ECP2M" | "XO2" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM2"	"ECP2"
pmi_byte_size	8 | 9	9

Note

1. For SC, SCM, ECP2, ECP2M, XP2, and ECP3 FPGA device families, the read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
2. For ECP3, "async" is not a valid option.

RAM_DP_TRUE (True Dual Port RAM)

Implementation: EBR

Table 306: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	DataInA	Bus	(pmi_data_width_a – 1) : 0
I	DataInB	Bus	(pmi_data_width_b – 1) : 0
I	AddressA	Bus	(pmi_addr_width_a – 1) : 0
I	AddressB	Bus	(pmi_addr_width_b – 1) : 0
I	ClockA	Bit	N/A
I	ClockB	Bit	N/A
I	ClockEnA	Bit	N/A
I	ClockEnB	Bit	N/A
I	WrA	Bit	N/A
I	WrB	Bit	N/A
I	ResetA	Bit	N/A
I	ResetB	Bit	N/A
O	QA	Bus	(pmi_data_width_a – 1) : 0
O	QB	Bus	(pmi_data_width_b – 1) : 0

Table 307: Parameters

Name	Value	Default
pmi_addr_depth_a[1]	2 to 65536	512
pmi_addr_width_a[1]	1 to 16	9
pmi_data_width_a	1 to 256	18
pmi_addr_depth_b[1]	2 to 65536	512
pmi_addr_width_b[1]	1 to 16	9
pmi_data_width_b	1 to 256	18
pmi_regmode_a	"reg" | "noreg"	"reg"
pmi_regmode_b	"reg" | "noreg"	"reg"
pmi_gsr	"enable" | "disable"	"disable"
pmi_resetmode[2]	"async" | "sync"	"sync"
pmi_optimization	"area" | "speed"	"speed"
pmi_init_file	<*string*>	"none"
pmi_init_file_format	"binary" | "hex"	"binary"
pmi_write_mode_a[3]	"normal" | "writethrough" | "readbeforewrite"	"normal"
pmi_write_mode_b[3]	"normal" | "writethrough" | "readbeforewrite"	"normal"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Note

1. For SC, SCM, ECP2, ECP2M, XP2, and ECP3 FPGA device families, the read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
2. For ECP3, "async" is not a valid option.
3. The "Readbeforewrite" option is not supported by ECP2, ECP2M, XP2, SC, SCM, or ECP3.

RAM_DP_TRUE_BE (True Dual Port RAM with Byte Enable)

Implementation: EBR

Table 308: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	DataInA	Bus	(pmi_data_width_a – 1) : 0
I	DataInB	Bus	(pmi_data_width_b – 1) : 0
I	AddressA	Bus	(pmi_addr_width_a – 1) : 0
I	AddressB	Bus	(pmi_addr_width_b – 1) : 0
I	ClockA	Bit	N/A
I	ClockB	Bit	N/A
I	ClockEnA	Bit	N/A
I	ClockEnB	Bit	N/A
I	WrA	Bit	N/A
I	WrB	Bit	N/A
I	ResetA	Bit	N/A
I	ResetB	Bit	N/A
O	QA	Bus	(pmi_data_width_a – 1) : 0
O	QB	Bus	(pmi_data_width_b – 1) : 0
I	ByteEnA	Bus	((pmi_data_width_a + pmi_byte_size – 1) / pmi_byte_size – 1) : 0
I	ByteEnB	Bus	((pmi_data_width_b + pmi_byte_size – 1) / pmi_byte_size – 1) : 0

Table 309: Parameters

Name	Value	Default
pmi_addr_depth_a[1]	2 to 65536	512
pmi_addr_width_a[1]	1 to 16	9
pmi_data_width_a	1 to 256	18
pmi_addr_depth_b[1]	2 to 65536	512
pmi_addr_width_b[1]	1 to 16	9
pmi_data_width_b	1 to 256	18
pmi_regmode_a	"reg" | "noreg"	"reg"

Table 309: Parameters (Continued)

Name	Value	Default
pmi_regmode_b	"reg" | "noreg"	"reg"
pmi_gsr	"enable" | "disable"	"disable"
pmi_resetmode[2]	"async" | "sync"	"sync"
pmi_optimization	"area" | "speed"	"speed"
pmi_init_file	*<string>*	"none"
pmi_init_file_format	"binary" | "hex"	"binary"
pmi_write_mode_a[3]	"normal" | "writethrough" | "readbeforewrite"	"normal"
pmi_write_mode_b[3]	"normal" | "writethrough" | "readbeforewrite"	"normal"
pmi_family	"XP2" | "SC" | "SCM" | "ECP2" | "ECP2M" | "XO2" | "LPTM2" | "ECP3" | "ECP5U" | "ECP5UM"	"ECP2"
pmi_byte_size	8 | 9	9

Note

1. For SC, SCM, ECP2, ECP2M, XP2, and ECP3 FPGA device families, the read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
2. For ECP3, "async" is not a valid option.
3. The "Readbeforewrite" option is not supported by ECP2, ECP2M, XP2, SC, SCM, or ECP3.

RAM_DQ (Single Port RAM)

Implementation: EBR

Table 310: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width – 1) : 0
I	Address	Bus	(pmi_addr_width – 1) : 0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WE	Bit	N/A
O	Q	Bus	(pmi_data_width – 1) : 0

Table 311: Parameter

Name	Value	Default
pmi_addr_depth[1]	2 to 65536	512
pmi_addr_width[1]	1 to 16	9
pmi_data_width	1 to 256	18
pmi_regmode	"reg" | "noreg"	"reg"
pmi_gsr	"enable" | "disable"	"disable"
pmi_resetmode[2]	"async" | "sync"	"sync"
pmi_optimization	"area" | "speed"	"speed"
pmi_init_file	*<string>*	"none"
pmi_init_file_format	"binary" | "hex"	"binary"
pmi_write_mode[3]	"normal" | "writethrough" | "readbeforewrite"	"normal"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "LPTM" | "LPTM2"	"EC"

Note

1. For SC, SCM, ECP2, ECP2M, XP2, and ECP3 FPGA device families, the read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
2. For ECP3, "async" is not a valid option.
3. The "Readbeforewrite" option is not supported by ECP2, ECP2M, XP2, SC, SCM, or ECP3.

RAM_DQ_BE (Single Port RAM with Byte Enable)

Implementation: EBR

Table 312: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width – 1) : 0
I	Address	Bus	(pmi_addr_width – 1) : 0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	Reset	Bit	N/A
I	WE	Bit	N/A

Table 312: Pins

Input/Output	Port Name	Type	Size (Buses Only)
O	Q	Bus	(pmi_data_width – 1) : 0
I	ByteEn	Bus	((pmi_data_width + pmi_byte_size – 1) / pmi_byte_size – 1) : 0

Table 313: Parameters

Name	Value	Default
pmi_addr_depth[1]	2 to 65536	512
pmi_addr_width[1]	1 to 16	9
pmi_data_width	1 to 256	18
pmi_regmode	"reg" | "noreg"	"reg"
pmi_gsr	"enable" | "disable"	"disable"
pmi_resetmode[2]	"async" | "sync"	"sync"
pmi_optimization	"area" | "speed"	"speed"
pmi_init_file	*<string>*	"none"
pmi_init_file_format	"binary" | "hex"	"binary"
pmi_write_mode[3]	"normal" | "writethrough" | "readbeforewrite"	"normal"
pmi_family	"XP2" | "SC" | "SCM" | "ECP2" | "ECP2M" | "XO2" | "LPTM2" | "ECP3" | "ECP5U"	"ECP2"
pmi_byte_size	8 | 9	9

Note

1. For SC, SCM, ECP2, ECP2M, XP2, and ECP3 FPGA device families, the read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
2. For ECP3, "async" is not a valid option.
3. The "Readbeforewrite" option is not supported by ECP2, ECP2M, XP2, SC, SCM, or ECP3.

ROM (Read Only Memory)

Implementation: EBR

Table 314: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width – 1) : 0
I	OutClock	Bit	N/A
I	OutClockEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width – 1) : 0

Table 315: Parameters

Name	Value	Default
pmi_addr_depth[1]	2 to 65536	512
pmi_addr_width[1]	1 to 16	9
pmi_data_width	1 to 256	18
pmi_regmode	"reg" | "noreg"	"reg"
pmi_gsr	"enable" | "disable"	"disable"
pmi_resetmode[2]	"async" | "sync"	"sync"
pmi_optimization	"area" | "speed"	"speed"
pmi_init_file	<*string*>	"none"
pmi_init_file_format	"binary" | "hex"	"binary"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Note

1. For SC, SCM, ECP2, ECP2M, XP2, and ECP3 FPGA device families, the read/write address depth ranges from 2 to 131072. The address width ranges from 1 to 17.
2. For ECP3, "async" is not a valid option.

Distributed_DPRAM (Distributed Dual Port RAM)

Implementation: LUT

Table 316: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	WrAddress	Bus	(pmi_addr_width – 1) : 0
I	Data	Bus	(pmi_data_width – 1) : 0
I	WrClock	Bit	N/A
I	WrClockEn	Bit	N/A
I	WE	Bit	N/A
I	RdAddress	Bus	(pmi_addr_width – 1) : 0
I	RdClock	Bit	N/A
I	RdClockEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width – 1) : 0

Table 317: Parameters

Name	Value	Default
pmi_addr_depth	2 to 8192	32
pmi_addr_width	1 to 13	5
pmi_data_width	1 to 256	8
pmi_regmode	"reg" | "noreg"	"reg"
pmi_init_file[1]	*<string>*	"none"
pmi_init_file_format[1]	"binary" | "hex"	"binary"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Note

1. The pmi_init_file and pmi_init_file_format parameters are not supported for EC, XP, ECP, ECP2, ECP2M, LPTM, and XO device families in distributed mode.

Distributed_ROM (Distributed Read Only Memory)

Implementation: LUT

Table 318: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width – 1) : 0
I	OutClock	Bit	N/A
I	OutClockEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width – 1) : 0

Table 319: Parameters

Name	Value	Default
pmi_addr_depth	2 to 8192	32
pmi_addr_width	1 to 13	5
pmi_data_width	1 to 128	8
pmi_regmode	"reg" | "noreg"	"reg"
pmi_init_file	*<string>*	"none"
pmi_init_file_format	"binary" | "hex"	"binary"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Distributed_SPRAM (Distributed Single Port RAM)

Implementation: LUT

Table 320: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Address	Bus	(pmi_addr_width – 1) : 0
I	Data	Bus	(pmi_data_width – 1) : 0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
I	WE	Bit	N/A

Table 320: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width – 1) : 0

Table 321: Parameters

Name	Value	Default
pmi_addr_depth	2 to 8192	32
pmi_addr_width	1 to 13	5
pmi_data_width	1 to 128	8
pmi_regmode	"reg" | "noreg"	"reg"
pmi_init_file[1]	*<string>*	"none"
pmi_init_file_format[1]	"binary" | "hex"	"binary"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Note

1. The pmi_init_file and pmi_init_file_format parameters are not supported for EC, XP, ECP, ECP2, ECP2M, LPTM, and XO device families in distributed mode.

FIFO (First In First Out Single Clock)

Implementation: EBR, LUT

Table 322: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width – 1) : 0
I	Clock	Bit	N/A
I	WrEn	Bit	N/A
I	RdEn	Bit	N/A
I	Reset	Bit	N/A
O	Q	Bus	(pmi_data_width – 1) : 0
O	Empty	Bit	N/A
O	Full	Bit	N/A

Table 322: Pins

Input/Output	Port Name	Type	Size (Buses Only)
O	AlmostEmpty	Bit	N/A
O	AlmostFull	Bit	N/A

Table 323: Parameters

Name	Value	Default
pmi_data_depth[1]	2 to 65536	256
pmi_data_width	1 to 256	8
pmi_almost_empty_flag	1 to 512	4
pmi_almost_full_flag	1 to 512	252
pmi_full_flag	1 to pmi_data_depth	256
pmi_empty_flag	0	0
pmi_regmode	"reg" | "noreg" | "outreg" | "outreg_rden"	"reg"
pmi_implementation	"EBR" | "LUT"	"EBR"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Note

1. The device depth for the LUT based FIFO ranges from 2 to 8192. The XP2, ECP2, ECP2M, SC, SCM, and ECP3 device family address depths range from 2 to 131072. The EC, ECP, and XP families range from 2 to 65536.

FIFO_DC (First In First Out Dual Clock)

Implementation: EBR, LUT

Table 324: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Data	Bus	(pmi_data_width_w – 1) : 0
I	WrClock	Bit	N/A
I	RdClock	Bit	N/A
I	WrEn	Bit	N/A
I	RdEn	Bit	N/A

Table 324: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Reset	Bit	N/A
I	RPReset	Bit	N/A
O	Q	Bus	(pmi_data_width_r – 1) : 0
O	Empty	Bit	N/A
O	Full	Bit	N/A
O	AlmostEmpty	Bit	N/A
O	AlmostFull	Bit	N/A

Table 325: Parameters

Name	Value	Default
pmi_data_depth_w[1]	2 to 131072	256
pmi_data_depth_r[1]	2 to 131072	256
pmi_data_width_w	1 to 256	18
pmi_data_width_r	1 to 256	18
pmi_full_flag	1 to pmi_data_depth_r	256
pmi_empty_flag	0	0
pmi_almost_full_flag	1 to pmi_data_depth_w	252
pmi_almost_empty_flag	1 to pmi_data_depth_r	4
pmi_regmode	"reg" | "noreg" | "outreg" | "outreg_rden"	"reg"
pmi_resetmode[2]	"async" | "sync"	"async"
pmi_implementation	"EBR" | "LUT"	"EBR"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "LPTM" | "LPTM2"	"EC"

Note

1. The device depth for the LUT-based FIFO_DC ranges from 2 to 8192. The XP2, ECP2, ECP2M, SC, SCM, and ECP3 device family data depths range from 2 to 131072. The EC, ECP, and XP families range from 2 to 65536. The XO, LPTM, LPTM2 and XO2 data depths range from 2 to 16384. The SC, SCM, XO, XO2, LPTM2, and LPTM device families support different read/write depths and different read/write data widths. The depth settings of SC and SCM are 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65535, and 131072. The depth settings for XO and LPTM are 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, and 16384.
2. The SC, SCM, XO, XO2, LPTM2 and LPTM device families support the pmi_resetmode parameter. The pmi_resetmode applies to the EBR memory implementation.

Shift Registers (Distributed RAM Shift Register)

Implementation: LUT, EBR[1]

Table 326: Pins

Input/Output	Port Name	Type	Size (Buses Only)
I	Din	Bus	(pmi_data_width – 1) : 0
I	Addr	Bus	(pmi_max_width – 1) : 0
I	Clock	Bit	N/A
I	ClockEn	Bit	N/A
O	Q	Bus	(pmi_data_width – 1) :0

Table 327: Parameters

Name	Value	Default
pmi_data_width	1 to 256	16
pmi_regmode	"reg" | "noreg"	"reg"
pmi_shiftreg_type[2]	"fixed" | "variable" | "lossless"	"fixed"
pmi_num_shift	2 to 1024	16
pmi_num_width	0 to 10	4
pmi_max_shift	2 to 1024	16
pmi_max_width	0 to 10	4
pmi_init_file[3]	<*string*>	"none"

Table 327: Parameters

Name	Value	Default
pmi_init_file_format[3]	"binary" | "hex"	"binary"
pmi_family	"EC" | "XP" | "XP2" | "SC" | "SCM" | "ECP" | "ECP2" | "ECP2M" | "XO" | "XO2" | "XO3L" | "XO3LF" | "ECP3" | "ECP5U" | "ECP5UM" | "LPTM" | "LPTM2"	"EC"

Note

1. EBR implementation is not available in ispLEVER 6.0 or earlier.
2. The lossless mode is not supported in ispLEVER 6.0 or earlier.
3. The pmi_init_file and pmi_init_file_format parameters are not supported for EC, XP, ECP, ECP2, ECP2M, LPTM, and XO device families in distributed mode.

Primitive Library - ECP5

This library includes compatible FPGA primitives supported by the ECP5 device family.

- Adder Subtractors
- Flip-Flops
- Input/Output Buffers
- ECP5 Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexers
- Multipliers in DSP Blocks
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read-Only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information on individual primitives, a variety of technical notes for the ECP5 family are available on the Lattice Web site.

- TN1260 - ECP5 sysCOFIG Usage Guide
- TN1261 - ECP5 SERDES/PCS Usage Guide
- TN1262 - ECP5 sysIO Usage Guide
- TN1263 - ECP5 sysCLOCK PLL/DLL Design and Usage Guide
- TN1264 - ECP5 Memory Usage Guide
- TN1265 - ECP5 High-Speed I/O Interface
- TN1266 - ECP5 Power Consumption and Management
- TN1267 - ECP5 sysDSP Usage Guide
- TN1184 - LatticeECP3 and ECP5 Soft Error Detection (SED) Usage Guide
- TN1269 - ECP5 Hardware Checklist

Table 328: Adder Subtractors

PRADD18A	18-Bit Pre Adder for DSP
PRADD9A	9 Bit Pre Adder for DSP

Table 329: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable

Table 329: Flip-Flops (Continued)

FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 330: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
IB	CMOS Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OLVDS	LVDS Output Buffer

Table 331: ECP5 Memory Primitives

DP16KD	True Dual Port Block RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
PDPW16KD	Pseudo Dual Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 332: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 333: Miscellaneous Logic

INV	Inverter

Table 333: Miscellaneous Logic

VHI	Logic High Generator
VLO	Logic Low Generator

Table 334: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 335: Multipliers in DSP Blocks

MULT18X18C	18x18 Multiplier in DSP blocks
MULT18X18D	DSP Multiplier
MULT9X9C	9x9 Multiplier Multipliers in DSP blocks
MULT9X9D	DSP Multiplier

PIC Cells

Table 336: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 337: PIC Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 338: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 339: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory
ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 340: Clock Manager/PLL/DLL

CLKDIVF	Clock Divider
DCCA	Dynamic Clock Control Block
DCSC	Dynamic Clock Selection
DLLDELD	Slave Delay
ECLKBRIDGECS	ECLK Bridge Block Clock Select
EHXPLLL	GPLL for ECP5.
OSCG	Oscillator for Configuration Clock
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 341: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 342: Dual Data Rate Cells

DDRDLLA	90 degree delay for the DQS Input during a memory interface or the clock input for a generic DDR interface
DQSBUFM	DQS circuit for DDR Memory.
ECLKSYNCB	ECLK Stop Block
IDDR71B	7:1 LVDS Input Supporting 1:7 Gearing
IDDRX1F	Generic Input DDR primitive
IDDRX2DQA	Implements DDR2 memory input interface at higher speeds and DDR3 memory interface.
IDDRX2F	Generic Input DDR primitive
ODDRX1F	Generic X1 ODDR implementation.
ODDRX2F	Generic X2 ODDR implementation.
ODDRX2DQA	Memory Output DDR Primitive for DQ outputs.
ODDRX2DQSB	Generates DQS clock output for DDR2 and DDR3 memory.
ODDR71B	7:1 LVDS ODDR implementation
OSHX2A	Generates the address and command for DDR3 memory with X2 gearing and write leveling.
TSHX2DQA	Generates the tristate control for DQ data output for DDR2 memory with X2 gearing and DDR3 memory.
TSHX2DQSA	Generate the tristate control for DQS output.

Table 343: Miscellaneous

ALU24A	24 Bit Ternary Adder/Subtractor
ALU24B	24-bit Ternary Adder/Subtractor for 9x9 Mode
ALU54A	54 Bit Ternary Adder/Subtractor
ALU54B	54 Bit Ternary Adder/Subtractor for Highspeed
BCINRD	Dynamic Bank Controller InRD
BCLVDSOB	Dynamic Bank Controller LVDS
CCU2C	Carry-Chain
DCUA	Dual Channel Unit Sci Interface
DELAYF	Delay
DELAYG	Delay
DTR	Digital temperature readout

Table 343: Miscellaneous (Continued)

EXTREFB	External Reference Clock
GSR	Global Set/Reset
IMIPI	Special Primitive for MIPI Input Support
INRDB	Input Reference and Differential Buffer
LVDSOB	LVDS Output Buffer
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PUR	Power Up Set/Reset
SEDGA	Soft Error Detect
SGSR	Synchronous Release Global Set/Reset Interface
START	Startup Controller
USRMCLK	Primitive to allow the user function to access the SPI PROM

Primitive Library - LatticeECP/EC and LatticeXP

This library includes compatible FPGA primitives supported by the LatticeXP and LatticeECP/EC device families.

- Adders Subtractors
- Comparators
- Counters
- Loadable Counters
- Flip-Flops
- Input/Output Buffers
- Latches
- LatticeECP DSP Block
- LatticeXP and LatticeEC Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexer
- Multipliers (Not DSP)
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read Only memory
- Special Cells
  - Clock/PLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information, a variety of technical documents for the LatticeECP/EC family and LatticeXP family are available on the Lattice Web site.

- TN1049 - LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide
- TN1050 - LatticeECP/EC and XP DDR Usage Guide
- TN1051 - Memory Usage Guide for LatticeECP/EC and LatticeXP Devices
- TN1052 - Estimating Power Using Power Calculator for LatticeECP/EC and LatticeXP Devices
- TN1053 - LatticeECP/EC sysCONFIG Usage Guide

- TN1056 - LatticeECP/EC and LatticeXP sysIO Usage Guide
- TN1057 - LatticeECP sysDSP Usage Guide
- TN1082 - LatticeXP sysCONFIG Usage Guide

Table 344: Adders Subtractors

FADD2	2 Bit Fast Adder
FSUB2	2 Bit Fast Subtractor (two's complement)
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)

Table 345: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 346: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 347: Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 347: Loadable Counters (Continued)

LB4P3AX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB4P3AY	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB4P3BX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB4P3DX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB4P3IX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB4P3JX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD2P3AY	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD4P3AX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD4P3AY	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD4P3BX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD4P3DX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD4P3IX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD4P3JX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear

Table 347: Loadable Counters (Continued)

LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU4P3AX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU4P3AY	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU4P3BX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU4P3DX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU4P3IX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU4P3JX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 348: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 348: Flip-Flops (Continued)

FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 349: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate and Pull-down BiDirectional
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate and Pull-up BiDirectional

Table 349: Input/Output Buffers

BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	CMOS Input Buffer
IBM	CMOS Input Buffer
IBMPD	CMOS Input Buffer with Pull-down
IBMPDS	CMOS Input Buffer with Pull-down and Delay
IBMPU	CMOS Input Buffer with Pull-up
IBMPUS	CMOS Input Buffer with Pull-up and Delay
IBMS	CMOS Input Buffer with Delay
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	6mA Sink 3mA Source Sinklim Output Buffer
OBZPD	12mA Sink 6mA Source Slewlim Output Buffer
OBZPU	12mA Sink 6mA Source Fast Output Buffer
OBW	6mA Sink 3mA Source Sinklim Output Buffer with Tristate
OLVDS	LVDS Output Buffer

Table 350: Latches

FD1S1A	Positive Level Data Latch with GSR Used for Clear
FD1S1AY	Positive Level Data Latch with GSR Used for Preset
FD1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset
FD1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear
FD1S1I	Positive Level Data Latch with Positive Level Synchronous Clear
FD1S1J	Positive Level Data Latch with Positive Level Synchronous Preset
FL1S1A	Positive Level Loadable Latch with Positive Select and GSR Used for Clear
FL1S1AY	Positive Level Loadable Latch with Positive Select and GSR Used for Preset
FL1S1B	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Preset

Table 350: Latches (Continued)

FL1S1D	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Clear
FL1S1I	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Clear
FL1S1J	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Preset

Table 351: LatticeECP DSP Block

MULT18X18	ECP 18X18 DSP Multiplier
MULT18X18ADDSUB	ECP 18X18 DSP Adder/Subtractor
MULT18X18ADDSUBSUM	ECP 18X18 DSP Adder/Subtractor/Sum
MULT18X18MAC	ECP 18X18 DSP MAC
MULT36X36	ECP 36X36 DSP Multiplier
MULT9X9	ECP 9X9 DSP Multiplier
MULT9X9ADDSUB	ECP 9X9 DSP Adder/Subtractor
MULT9X9ADDSUBSUM	ECP 9X9 DSP Adder/Subtractor/Sum
MULT9X9MAC	ECP 9X9 DSP MAC

Table 352: LatticeXP and LatticeEC Memory Primitives

DP8KA	8K Dual Port Block RAM
DPR16X2B	16 Word by 2 Dual Port RAM (within PFU)
PDP8KA	8K Pseudo Dual Port Block RAM
SP8KA	8K Single Port Block RAM
SPR16X2B	16 Word by 2 Single Port RAM (within PFU)

Table 353: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate

Table 353: Logic Gates (Continued)

ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR11	11 Input Exclusive OR Gate
XOR2	2 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate

Table 354: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 355: Multiplexer

L6MUX21	2 to 1 Mux
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux

Table 355: Multiplexer

MUX321	32-Input Mux within the PFU (8 Slices)
MUX4	4-bit Multiplexer
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 356: Multipliers (Not DSP)

MULT2	2X2 Multiplier

PIC Cells

Table 357: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 358: PIC Flip-Flops (Output)

OFE1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and Edge Clock (used in output PIC area only)
OFE1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and Edge Clock (used in output PIC area only)
OFE1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and Edge Clock (used in output PIC area only)
OFE1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and Edge Clock (used in output PIC area only)

Table 358: PIC Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 359: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 360: Read Only memory

ROM16X1	16 Word by 1 Bit Read-Only Memory
ROM32X1	32 Word by 1 Bit Read-Only Memory
ROM64X1	64 Word by 1 Bit Read-Only Memory
ROM128X1	128 Word by 1 Bit Read-Only Memory
ROM256X1	256 Word by 1 Bit Read-Only Memory

Special Cells

Table 361: Clock/PLL

DCS	Dynamic Clock Selection Multiplexer

Table 361: Clock/PLL

EPLLB	Enhanced PLL
EHXPLLB	Enhanced High Performance with Dynamic Input Delay Control PLL

Table 362: Combinatorial Primitives

ORCALUT4	4-Input Look Up Table
ORCALUT5	5-Input Look Up Table
ORCALUT6	6-Input Look Up Table
ORCALUT7	7-Input Look Up Table
ORCALUT8	8-Input Look Up Table

Table 363: Dual Data Rate Cells

DQSBUFB	DDR DQS Buffer used as DDR memory DQS generator
DQSDLL	DLL used as DDR memory DQS DLL
IDDRXB	DDRX Input Cell
ODDRXB	Output DDR

Table 364: Miscellaneous

CCU2	Carry Chain
DELAY	Delay
GSR	Global Set/Reset
IBDDC	Dynamic Delay
JTAGB	JTAG (Joint Test Action Group) Controller
JTAGG	JTAG (Joint Test Action Group) Controller
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PUR	Power Up Set/Reset
SGSR	Synchronous Release Global Set/Reset Interface
STRTUP	Startup Controller

Primitive Library - LatticeECP2/M

This library includes compatible FPGA primitives supported by the LatticeECP2/M (including the “S-Series” LatticeECP2S and LatticeECP2MS) device families.

- Adders/Subtractors
- Comparators
- Counters
- Loadable Counters
- Flip-Flops
- Input/Output Buffer
- LatticeECP2/M Memory Primitive
- Logic Gates
- Miscellaneous Logic
- Multiplexers
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read-only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information, a variety of technical notes for the LatticeECP2/M family are available on the Lattice Web site.

- TN1102 - LatticeECP2/M sysIO Usage Guide
- TN1103 - LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide
- TN1104 - LatticeECP2/M Memory Usage Guide
- TN1105 - LatticeECP2/M High-Speed I/O Interface
- TN1106 - LatticeECP2/M Power Estimation and Management
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1108 - LatticeECP2/M sysCONFIG Usage Guide
- TN1109 - LatticeECP2/M Configuration Encryption Usage Guide
- TN1113 - LatticeECP2/M Soft Error Detection (SED) Usage Guide
- TN1124 - LatticeECP2M SERDES/PCS Usage Guide

Table 365: Adders/Subtractors

FADD2B	2 Bit Fast Adders/Subtractors
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)
FSUB2B	2 Bit Subtractor

Table 366: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 367: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 368: Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear

Table 368: Loadable Counters (Continued)

LD2P3AY	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 369: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 369: Flip-Flops (Continued)

FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 370: Input/Output Buffer

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up

Table 370: Input/Output Buffer (Continued)

BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	CMOS Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBW	Output Buffer with Tristate
OBZ	Output Buffer with Tristate
OBZPD	Output Buffer with Tristate and Pull-down
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 371: LatticeECP2/M Memory Primitive

DP16KB	True Dual Port Block RAM
DPR16X4A	Distributed Pseudo Dual Port RAM (within PFU)
PDPW16KB	Pseudo Dual Port Block RAM
SP16KB	Single Port Block RAM
SPR16X4A	Distributed Single Port RAM (within PFU)

Table 372: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate

Table 372: Logic Gates (Continued)

OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 373: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 374: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 375: Multipliers in DSP Blocks

MULT18X18ADDSUBB	18x18 Multiplier Add/Subtract Multipliers in DSP blocks
MULT18X18ADDSUBSUMB	18x18 Multiplier Add/Subtract and SUM Multipliers in DSP blocks
MULT18X18B	18x18 Multiplier in DSP blocks
MULT18X18MACB	18x18 Multiplier Accumulate Multipliers in DSP blocks
MULT36X36B	36x36 Multiplier Multipliers in DSP blocks
MULT9X9ADDSUBB	9x9 Multiplier Add/Subtract Multipliers in DSP blocks
MULT9X9ADDSUBSUMB	9x9 Multiplier Add/Subtract and SUM Multipliers in DSP blocks
MULT9X9B	9x9 Multiplier Multipliers in DSP blocks

PIC Cells

Table 376: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 377: PIC Flip-Flops (Output)

OFE1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and Edge Clock (used in output PIC area only)
OFE1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and Edge Clock (used in output PIC area only)
OFE1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and Edge Clock (used in output PIC area only)

Table 377: PIC Flip-Flops (Output) (Continued)

OFE1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and Edge Clock (used in output PIC area only)
OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 378: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 379: Read-only Memory

ROM128X1	128 Word by 1 bit read-only memory
ROM16X1	16 Word by 1 bit read-only memory
ROM256X1	256 Word by 1 bit read-only memory
ROM32X1	32 Word by 1 bit read-only memory
ROM64X1	64 Word by 1 bit read-only memory

Special Cells

Table 380: Clock Manager/PLL/DLL

CIDDLLA	Clock Injection Delay Removal
CLKDIVB	Clock Divider
DCS	Dynamic Clock Selection Multiplexer
DLLDELA	Slave Delay
EHXPLLD	Complex PLL
EPLLD	Enhanced PLL
OSCD	Oscillator for configuration clock
TRDLLA	Time Reference Delay

Table 381: Combinatorial Primitives

ORCALUT4	4-Input Look Up Table
ORCALUT5	5-Input Look Up Table
ORCALUT6	6-Input Look Up Table
ORCALUT7	7-Input Look Up Table
ORCALUT8	8-Input Look Up Table

Table 382: Dual Data Rate Cells

DQSBUFC	DQS Delay Function and Clock Polarity Selection Logic
DQSDLL	DLL Used as DDR Memory DQS DLL
IDDRFXA	DDR Generic Input with Full Clock Transfer (x1 Gearbox)
IDDRMFX1A	DDR Input and DQS to System Clock Transfer Registers with Full Clock Cycle Transfer
IDDRMX1A	DDR Input and DQS to System Clock Transfer Registers with Half Clock Cycle Transfer
IDDRX2B	DDR Generic Input with 2x Gearing Ratio
IDDRXC	DDR Generic Input
ODDRMXA	DDR Output Registers
ODDRX2B	DDR Generic Output with 2x Gearing Ratio
ODDRXC	DDR Generic Output

Table 383: Miscellaneous

CCU2B	Carry-Chain
DELAYB	Dynamic Delay in PIO
GSR	Global Set/Reset
JTAGC	JTAG (Joint Test Action Group) Controller
MULT2	2X2 Multiplier (not DSP)
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PUR	Power Up Set/Reset
SEDAA	SED BASIC
SGSR	Synchronous Release Global Set/Reset Interface
SPIM	SPIM Primitive
STRTUP	Startup Controller

Primitive Library - LatticeECP3

This library includes compatible FPGA primitives supported by the LatticeECP3 device family.

- Adders/Subtractors
- Comparators
- Counters
- Loadable Counters
- Flip-Flops
- Input/Output Buffers
- LatticeECP3 Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexer
- Multipliers in DSP Blocks
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read-Only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information on individual primitives, a variety of technical notes for the LatticeECP3 family are available on the Lattice Web site.

- TN1177 - LatticeECP3 sysIO Usage Guide
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide
- TN1179 - LatticeECP3 Memory Usage Guide
- TN1180 - LatticeECP3 High-Speed I/O Interface
- TN1181 - Power Consumption and Management for LatticeECP3 Devices
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1169 - LatticeECP3 sysCONFIG Usage Guide
- TN1184 - LatticeECP3 and ECP5 Soft Error Detection (SED) Usage Guide
- TN1176 - LatticeECP3 SERDES/PCS Usage Guide

Table 384: Adders/Subtractors

FADD2B	2 Bit Fast Adders/Subtractors
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)
FSUB2B	2 Bit Subtractor

Table 385: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 386: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 387: Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear

Table 387: Loadable Counters (Continued)

LD2P3AY	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 388: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 388: Flip-Flops (Continued)

FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 389: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up

Table 389: Input/Output Buffers (Continued)

BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	CMOS Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 390: LatticeECP3 Memory Primitives

DP16KC	True Dual Port Block RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
PDPW16KC	Pseudo Dual Port Block RAM
SP16KC	Single Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 391: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate

Table 391: Logic Gates (Continued)

OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 392: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 393: Multiplexer

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 394: Multipliers in DSP Blocks

MULT18X18C	18x18 Multiplier in DSP blocks
MULT9X9C	9x9 Multiplier Multipliers in DSP blocks

PIC Cells

Table 395: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 396: PIC Flip-Flops (Output)

OFD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear. Used to Tri-State DDR/DDR2
OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 397: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 398: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory
ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 399: Clock Manager/PLL/DLL

CIDDLLB	Clock Injection Delay Removal
CLKDIVB	Clock Divider
DCCA	*(For internal use only)* Dynamic Quadrant Clock Enable/Disable
DCS	Dynamic Clock Selection Multiplexer
DLLDELB	Slave Delay
EHXPLLF	Complex PLL
OSCF	Oscillator for Configuration Clock
TR1DLLB	Time Reference DLL with Dynamic Delay Adjustment
TRDLLB	Time Reference DLL

Table 400: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 401: Dual Data Rate Cells

DQSBUFD	DDR DQS Buffer Used for DDR3_MEM and DDR3_MEMGEN
DQSBUFE	DDR DQS Buffer Used for DDR_GENX2
DQSBUFE1	DDR DQS Buffer Used for DDR_GENX2
DQSBUFF	DDR DQS Buffer Used for DDR_MEM, DDR2_MEM, and DDR2_MEMGEN
DQSBUFG	DDR DQS Buffer Used for DDR_GENX1
DQSDLLB	DQS DLL for DDR_MEM, DDR2_MEM, and DDR3_MEM
ECLKSYNCA	ECLK Stop Block
IDDRX2D	Input DDR for DDR3_MEM, DDR_GENX2, and DDR3_MEMGEN
IDDRX2D1	Input DDR for DDR_GENX2
IDDRXD	Input DDR for DDR_MEM, DDR2_MEM, DDR_GENX1, and DDR2_MEMGEN
IDDRXD1	Input DDR for DDR_GENX1
ODDRTDQA	Tri-State for DQ: DDR3_MEM and DDR_GENX2
ODDRTDQSA	Tri-State for Single-Ended and Differential DQS: DDR_MEM, DDR2_MEM, and DDR3_MEM
ODDRX2D	Output DDR for DDR3_MEM and DDR_GENX2
ODDRX2DQSA	Output for Differential DQS: DDR3_MEM
ODDRXD	Output DDR for DDR_MEM, DDR2_MEM, DDR_GENX1, and DDR2_MEMGEN
ODDRXD1	Output DDR for DDR_GENX1
ODDRXDQSA	Output for Single-Ended and Differential DQS: DDR_MEM, DDR2_MEM, and DDR2_MEMGEN

Table 402: Miscellaneous

ALU24A	24 Bit Ternary Adder/Subtractor
ALU54A	54 Bit Ternary Adder/Subtractor
CCU2C	Carry-Chain
DELAYB	Dynamic Delay in PIO
DELAYC	Fixed Delay in PIO
GSR	Global Set/Reset
JTAGE	JTAG (Joint Test Action Group) Controller
MULT2	2X2 Multiplier (not DSP)
PERREGA	Persistent User Register
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PUR	Power Up Set/Reset
SEDCA	Basic SED (Soft Error Detect)
SGSR	Synchronous Release Global Set/Reset Interface
SPIM	SPIM Primitive
START	Startup Controller

Primitive Library - LatticeSC/M

This library includes compatible FPGA primitives supported by the LatticeSC and LatticeSCM device families.

- Adders/Subtractors
- Comparators
- Counters
- Loadable Counters
- Flip-Flops
- Input/Output Buffers
- Latches
- LatticeSC/M Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexers
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Flip-Flops (Latched)
  - PIC Latches (Input)
  - PIC Shift Registers
- Read-Only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information, a variety of technical documents for the LatticeSC/M family are available on the Lattice Web site.

- TN1080 - LatticeSC sysCONFIG Usage Guide
- TN1085 - LatticeSC MPI/System Bus
- TN1088 - LatticeSC PURESPEED I/O Usage Guide
- TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices
- TN1096 - LatticeSC QDR-II SRAM Memory Interface User's Guide
- TN1098 - LatticeSC sysCLOCK and PLL/DLL User's Guide
- TN1099 - LatticeSC DDR/DDR2 SDRAM Memory Interface User's Guide
- TN1101 - Power Calculations and Considerations for LatticeSC Devices

Table 403: Adders/Subtractors

FADD2	2 Bit Fast Adder
FSUB2	2 Bit Fast Subtractor (two's complement)

Table 404: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 405: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CU2	Combinational Logic for 2 Bit Up Counter
CD2	Combinational Logic for 2 Bit Down Counter

Table 406: Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LB4P3AX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB4P3AY	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset

Table 406: Loadable Counters (Continued)

LB4P3BX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB4P3DX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB4P3IX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB4P3JX	4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD2P3AY	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD4P3AX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD4P3AY	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD4P3BX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD4P3DX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD4P3IX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD4P3JX	4 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset

Table 406: Loadable Counters (Continued)

LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU4P3AX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU4P3AY	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU4P3BX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU4P3DX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU4P3IX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU4P3JX	4 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 407: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset

Table 407: Flip-Flops (Continued)

FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 408: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	Input Buffer

Table 408: Input/Output Buffers (Continued)

IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBW	Output Buffer with Tristate
OBZ	Output Buffer with Tristate
OBZPD	Output Buffer with Tristate and Pull-down
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 409: Latches

FD1S1A	Positive Level Data Latch with GSR Used for Clear
FD1S1AY	Positive Level Data Latch with GSR Used for Preset
FD1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset
FD1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear
FD1S1I	Positive Level Data Latch with Positive Level Synchronous Clear
FD1S1J	Positive Level Data Latch with Positive Level Synchronous Preset
FL1S1A	Positive Level Loadable Latch with Positive Select and GSR Used for Clear
FL1S1AY	Positive Level Loadable Latch with Positive Select and GSR Used for Preset
FL1S1B	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Preset
FL1S1D	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Clear
FL1S1I	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Clear
FL1S1J	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Preset

Table 410: LatticeSC/M Memory Primitives

DP16KA	16K Dual Port Block RAM
DPR16X2	16 Word by 2 Distributed Dual Port RAM (within PFU)

Table 410: LatticeSC/M Memory Primitives (Continued)

FIFO16KA	16K FIFO
PDP16KA	16K Pseudo Dual Port Block RAM
SP16KA	16 Word by 16 Bit Single Port Block RAM
SPR16X2	16 Word by 2 Distributed Single Port RAM (within PFU)

Table 411: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 412: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 413: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

PIC Cells

Table 414: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 415: PIC Flip-Flops (Output)

OFE1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and Edge Clock (used in output PIC area only)
OFE1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and Edge Clock (used in output PIC area only)
OFE1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and Edge Clock (used in output PIC area only)
OFE1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and Edge Clock (used in output PIC area only)
OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 416: PIC Flip-Flops (Latched)

ILF2P3BX	Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, and Positive Level Asynchronous Preset (used in input PIC area only)
ILF2P3DX	Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, and Positive Level Asynchronous Clear (used in input PIC area only)
ILF2P3IX	Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, and Positive Level Synchronous Clear (Clear overrides Enable) (used in input PIC area only)
ILF2P3JX	Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, and Positive Level Synchronous Preset (Preset overrides Enable) (used in input PIC area only)

Table 417: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 418: PIC Shift Registers

ISRX1A	Input 1-Bit Shift Register
ISRX2A	Input 2-Bit Shift Register
ISRX4A	Input 4-Bit Shift Register
OSRX1A	Output 1-Bit Shift Register
OSRX2A	Output 2-Bit Shift Register
OSRX4A	Output 4-Bit Shift Register

Table 419: Read-Only Memory

ROM16X1	16 Word by 1 bit read-only memory
ROM32X1	32 Word by 1 bit read-only memory
ROM32X4	32 Word by 4 bit read-only memory
ROM64X1	64 Word by 1 bit read-only memory
ROM128X1	128 Word by 1 bit read-only memory
ROM256X1	256 Word by 1 bit read-only memory

Special Cells

Table 420: Clock Manager/PLL/DLL

CIDDLLA	Clock Injection Delay Removal
CIMDLLA	Clock Injection Match
CLKCNTL	Clock Controller
CLKDET	Clock Detect
CLKDIV	Clock Divider
DCS	Dynamic Clock Selection Multiplexer
EHXPLLA	Enhanced High Performance with Dynamic Input Delay Control PLL
OSCA	Internal Oscillator
SDCDLLA	Single Delay Cell DLL
TRDLLA	Time Reference Delay

Table 421: Dual Data Rate Cells

IDDRA	Input DDR
IDDRX1A	Input DDR
IDDRX2A	Input DDR
IDDRX4A	Input DDR
ODDRA	Output DDR
ODDRXA	Output DDR
ODDRX2A	Output DDR
ODDRX4A	Output DDR

Table 422: Miscellaneous

DELAY	Delay
GSR	Global Set/Reset
JTAGA	JTAG Logic Control Circuit
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PUR	Power Up Set/Reset
PVTIOCTRL	PVT Monitor Circuit Controller

Table 422: Miscellaneous (Continued)

RDBK	Readback Controller
SGSR	Synchronous Release Global Set/Reset Interface
STRTUP	Startup Controller
TSALL	Global Tristate Interface

Primitive Library - LatticeXP2

This library includes compatible FPGA primitives supported by the LatticeXP2 device family

- Arithmetic Functions
- Comparators
- Counters
- Loadable Counters
- Flip-Flops
- Input/Output Buffers
- LatticeXP2 Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexers
- Multipliers in DSP Blocks
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read-Only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information, a variety of technical notes for the LatticeXP2 family are available on the Lattice web site.

- TN1142 - LatticeXP2 Configuration Encryption and Security Usage Guide
- TN1138 - LatticeXP2 High-Speed I/O Interface
- TN1137 - LatticeXP2 Memory Usage Guide
- TN1130 - LatticeXP2 Soft Error Detection (SED) Usage Guide
- TN1126 - LatticeXP2 sysCLOCK PLL Design and Usage Guide
- TN1141 - LatticeXP2 sysCONFIG Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide
- TN1136 - LatticeXP2 sysIO Usage Guide
- TN1139 - Power Estimation and Management for LatticeXP2 Devices

Table 423: Arithmetic Functions

FADD2B	2 Bit Fast Adders/Subtractors
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)
FSUB2B	2 Bit Subtractor
MULT2	2X2 Multiplier (not DSP)

Table 424: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 425: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 426: Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear

Table 426: Loadable Counters (Continued)

LD2P3AY	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 427: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Table 427: Flip-Flops (Continued)

FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 428: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up

Table 428: Input/Output Buffers (Continued)

BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	CMOS Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBW	Output Buffer with Tristate
OBZ	Output Buffer with Tristate
OBZPD	Output Buffer with Tristate and Pull-down
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 429: LatticeXP2 Memory Primitives

DP16KB	True Dual Port Block RAM
DPR16X4A	Distributed Pseudo Dual Port RAM (within PFU)
DPR16X4B	Distributed Pseudo Dual Port RAM (within PFU)
PDPW16KB	Pseudo Dual Port Block RAM
SP16KB	Single Port Block RAM
SPR16X4A	Distributed Single Port RAM (within PFU)
SPR16X4B	Distributed Single Port RAM (within PFU)
SSPIA	SSPI TAG Memory
STFA	Store to Flash Primitive

Table 430: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate

Table 430: Logic Gates (Continued)

ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 431: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 432: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux

Table 432: Multiplexers (Continued)

MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 433: Multipliers in DSP Blocks

MULT18X18ADDSUBB	18x18 Multiplier Add/Subtract Multipliers in DSP blocks
MULT18X18ADDSUBSUMB	18x18 Multiplier Add/Subtract and SUM Multipliers in DSP blocks
MULT18X18B	18x18 Multiplier in DSP blocks
MULT18X18MACB	18x18 Multiplier Accumulate Multipliers in DSP blocks
MULT36X36B	36x36 Multiplier Multipliers in DSP blocks
MULT9X9ADDSUBB	9x9 Multiplier Add/Subtract Multipliers in DSP blocks
MULT9X9ADDSUBSUMB	9x9 Multiplier Add/Subtract and SUM Multipliers in DSP blocks
MULT9X9B	9x9 Multiplier Multipliers in DSP blocks

PIC Cells

Table 434: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 435: PIC Flip-Flops (Output)

OFE1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and Edge Clock (used in output PIC area only)
OFE1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and Edge Clock (used in output PIC area only)
OFE1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and Edge Clock (used in output PIC area only)
OFE1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and Edge Clock (used in output PIC area only)
OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 436: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 437: Read-Only Memory

ROM16X1	16 Word by 1 bit read-only memory
ROM32X1	32 Word by 1 bit read-only memory
ROM64X1	64 Word by 1 bit read-only memory
ROM128X1	128 Word by 1 bit read-only memory
ROM256X1	256 Word by 1 bit read-only memory

Special Cells

Table 438: Clock Manager/PLL/DLL

CLKDIVB	Clock Divider
DCS	Dynamic Clock Selection Multiplexer
EHXPLLE	Complex PLL
EHXPLLE1	Complex PLL
EPLLD	Enhanced PLL
EPLLD1	Enhanced PLL
OSCE	Oscillator for configuration clock

Table 439: Combinatorial Primitives

ORCALUT4	4-Input Look Up Table
ORCALUT5	5-Input Look Up Table
ORCALUT6	6-Input Look Up Table

Table 439: Combinatorial Primitives (Continued)

ORCALUT7	7-Input Look Up Table
ORCALUT8	8-Input Look Up Table

Table 440: Dual Data Rate Cells

DQSBUFC	DQS Delay Function and Clock Polarity Selection Logic
DQSDLL	DLL Used as DDR Memory DQS DLL
IDDRFXA	DDR Generic Input with Full Clock Transfer (x1 Gearbox)
IDDRMFX1A	DDR Input and DQS to System Clock Transfer Registers with Full Clock Cycle Transfer
IDDRMX1A	DDR Input and DQS to System Clock Transfer Registers with Half Clock Cycle Transfer
IDDRX2B	DDR Generic Input with 2x Gearing Ratio
IDDRXC	DDR Generic Input
ODDRMXA	DDR Output Registers
ODDRX2B	DDR Generic Output with 2x Gearing Ratio
ODDRXC	DDR Generic Output

Table 441: Miscellaneous

CCU2B	Carry-Chain
DELAYB	Dynamic Delay in PIO
GSR	Global Set/Reset
IOWAKEUPA	XP2 Wake-up Controller
JTAGE	JTAG (Joint Test Action Group) Controller
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PUR	Power Up Set/Reset
SEDBA	XP2 SED BASIC
SEDBB	XP2 SED BASIC for One Shot Mode
SGSR	Synchronous Release Global Set/Reset Interface
SSPIA	XP2 SSPI TAG Memory
START	Startup Controller
STFA	XP2 Store to Flash Primitive

Primitive Library - LFMNX

This library includes compatible primitives supported by the LFMNX (Mach-NX) device family.

- Adders/Subtractors
- Comparators
- Counters
  - Bi-Directional Loadable Counters
  - Loadable Down Counters
  - Loadable Up Counters
- Flip-Flops
- Input/Output Buffer
- Latches
- Logic Gates
- PIC Cells
  - Flip-Flops (Input)
  - Flip-Flops (Output)
  - PIC Latches (Input)
- Memory Primitives
- Miscellaneous Logic
- Multiplexers
- Multipliers
- Read-Only Memory
- Special Cells
  - Clock/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

Table 442: Adders/Subtractors

FADD2B	Fast 2 Bit Adder
FSUB2B	2 Bit Subtractor
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)

Table 443: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 444: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 445: Bi-Directional Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 446: Loadable Down Counters

LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD2P3AY	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 447: Loadable Up Counters

LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 448: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset

Table 448: Flip-Flops (Continued)

FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 449: Input/Output Buffer

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 450: Latches

FD1S1A	Positive Level Data Latch with GSR Used for Clear
FD1S1AY	Positive Level Data Latch with GSR Used for Preset
FD1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset
FD1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear
FD1S1I	Positive Level Data Latch with Positive Level Synchronous Clear
FD1S1J	Positive Level Data Latch with Positive Level Synchronous Preset
FL1S1A	Positive Level Loadable Latch with Positive Select and GSR Used for Clear
FL1S1AY	Positive Level Loadable Latch with Positive Select and GSR Used for Preset
FL1S1B	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Preset
FL1S1D	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Clear
FL1S1I	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Clear
FL1S1J	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Preset

Table 451: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate

Table 451: Logic Gates (Continued)

XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR11	11 Input Exclusive OR Gate
XOR2	2 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate

PIC Cells

Table 452: Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 453: Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 454: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 455: Memory Primitives

DP8KC	8K True Dual Port Block RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
FIFO8KB	8K FIFO Block RAM

Table 455: Memory Primitives

PDPW8KC	Pseudo Dual Port Block RAM
SP8KC	8K Single Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 456: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 457: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 458: Multipliers

MULT2	2x2 Multiplier

Table 459: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory
ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 460: Clock/PLL/DLL

CLKDIVC	Clock Divider
DCCA	Dynamic Quadrant Clock Enable/Disable
DCMA	Dynamic Clock Mux

Table 460: Clock/PLL/DLL

DLLDELC	Clock Shifting for ECLK or PCLK
ECLKBRIDGECS	ECLK Bridge Block Clock Select
ECLKSYNCA	ECLK Stop Block
EHXPLLJ	GPLL
OSCJ	Oscillator
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 461: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 462: Dual Data Rate Cells

DQSDLLC	Master DLL for Generating Required Delay
IDDRXE	Input for Generic DDR X1 Using 1:2 Gearing
IDDRX2E	Input for Generic DDR X2 Using 1:4 Gearing
IDDRX4B	Input for Generic DDR X4 Using 1:8 Gearing
IDDRX71A	7:1 LVDS Input Supporting 1:7 Gearing
ODDRXE	Output for Generic DDR X1 Using 2:1 Gearing
ODDRX2E	Output for Generic DDR X2 Using 4:1 Gearing
ODDRX4B	Output for Generic DDR X4 Using 8:1 Gearing
ODDRX71A	7:1 LVDS Output

Table 463: Miscellaneous

BCINRD	Dynamic Bank Controller InRD
BCLVDSO	Dynamic Bank Controller LVDS
BCSLEWRATEA	Bank Controller for Slew Rate
CCU2D	Carry Chain
CLKFBBUFA	Dummy Feedback Delay Between PLL clk Output and PLL fb Port
DELAYD	Dynamic Delay for Bottom Bank
DELAYE	Fixed Delay in PIO

Table 463: Miscellaneous (Continued)

EFBB	Embedded Function Block for MachXO3D
ESBA	Embedded Security Block
GSR	Global Set/Reset
INRDB	Input Reference and Differential Buffer
JTAGF	JTAG (Joint Test Action Group) Controller
LVDSOB	LVDS Output Buffer
PCNTR	Power Controller
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PG	Power Guard
PUR	Power Up Set/Reset
SEDFA	Soft Error Detect in Basic Mode
SEDFB	Soft Error Detect in One Shot Mode
SGSR	Synchronous Release Global Set/Reset Interface
START	Startup Controller
TSALL	Global Tristate Interface

Primitive Library - LIFMD

This library includes compatible FPGA primitives supported by the LIFMD device family.

- Flip-Flops
- Input/Output Buffers
- Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexers
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read-Only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Combinatorial Primitives
  - Miscellaneous

References

For further information on individual primitives, a variety of technical notes for the LIFMD family are available on the Lattice web site.

- TN1301 – CrossLink High-Speed I/O Interface
- TN1302 – CrossLink Hardware Checklist
- TN1303 – CrossLink Programming and Configuration Usage Guide
- TN1304 – CrossLink sysCLOCK PLL/DLL Design and Usage Guide
- TN1305 – CrossLink sysI/O Usage Guide
- TN1306 – CrossLink Memory Usage Guide
- TN1307 – Power Management and Calculation for CrossLink Devices
- TN1308 – CrossLink I2C Hardened IP Usage Guide
- TN1309 – Advanced CrossLink I2C Hardened IP Reference Guide

Table 464: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset

Table 464: Flip-Flops (Continued)

FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 465: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
IB	CMOS Input Buffer
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OLVDS	LVDS Output Buffer

Table 466: Memory Primitives

DP8KE	True Dual Port EBR RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
PDPW8KE	Pseudo Dual Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 467: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate

Table 467: Logic Gates

NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 468: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 469: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

PIC Cells

Table 470: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 471: PIC Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 472: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 473: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory

Table 473: Read-Only Memory

ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 474: Clock Manager/PLL/DLL

CLKDIVG	Clock Divider
DCCA	Dynamic Clock Control Block
DCSC	Dynamic Clock Selection
DLLDELD	Slave Delay
EHXPLLM	GPLL
OSCI	Oscillator for Configuration Clock
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 475: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 476: Dual Data Rate Cells

DDRDLLA	90-degree delay for the DQS Input during a memory interface or the clock input for a generic DDR interface
ECLKSYNCB	ECLK Stop Block
IDDR71B	7:1 LVDS Input Supporting 1:7 Gearing
IDDRX1F	Generic Input DDR primitive
IDDRX2F	Generic Input DDR primitive
IDDRX4C	DDR primitive
IDDR141A	DDR primitive
IDDRX8A	DDR primitive
ODDRX1F	Generic X1 ODDR implementation
ODDRX2F	Generic X2 ODDR implementation

Table 476: Dual Data Rate Cells

ODDRX4C	ODDR primitive
ODDR141A	ODDR primitive
ODDRX8A	ODDR primitive
ODDR71B	7:1 LVDS ODDR implementation

Table 477: Miscellaneous

BCINRD	Dynamic Bank Controller InRD
BCLVDSOB	Bank Controller for LVDS Outut Buffers
CCU2C	Carry-Chain
DELAYF	Delay
DELAYG	Delay
GSR	Global Set/Reset
INRDB	Input Reference and Differential Buffer
I2CA	I2C Primitive
LVDSOB	LVDS Output Buffer
MIPI	Special Primitive for MIPI Input Support
MIPIDPHYA	Primitive
PFUMX	2-Input mux within the PFU, C0 used for Selection with Positive Select
PMUA	Power Management Unit
PUR	Power Up Set/Reset
SGSR	Synchronous Release Global Set/Reset Interface

Primitive Library - LIFMDF

This library includes compatible FPGA primitives supported by the LIFMDF device family.

- Flip-Flops
- Input/Output Buffers
- Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexers
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read-Only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Combinatorial Primitives
  - Miscellaneous

References

For further information on individual primitives, a variety of technical notes for the LIFMD family are available on the Lattice web site.

- TN1301 – CrossLink High-Speed I/O Interface
- TN1302 – CrossLink Hardware Checklist
- TN1303 – CrossLink Programming and Configuration Usage Guide
- TN1304 – CrossLink sysCLOCK PLL/DLL Design and Usage Guide
- TN1305 – CrossLink sysI/O Usage Guide
- TN1306 – CrossLink Memory Usage Guide
- TN1307 – Power Management and Calculation for CrossLink Devices
- TN1308 – CrossLink I2C Hardened IP Usage Guide
- TN1309 – Advanced CrossLink I2C Hardened IP Reference Guide

Table 478: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset

Table 478: Flip-Flops (Continued)

FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 479: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
IB	CMOS Input Buffer
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OLVDS	LVDS Output Buffer

Table 480: Memory Primitives

DP8KE	True Dual Port EBR RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
PDPW8KE	Pseudo Dual Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 481: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate

Table 481: Logic Gates

NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 482: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 483: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

PIC Cells

Table 484: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 485: PIC Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 486: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 487: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory

Table 487: Read-Only Memory

ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 488: Clock Manager/PLL/DLL

CLKDIVG	Clock Divider
DCCA	Dynamic Clock Control Block
DCSC	Dynamic Clock Selection
DLLDELD	Slave Delay
EHXPLLM	GPLL
OSCI	Oscillator for Configuration Clock
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 489: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 490: Dual Data Rate Cells

DDRDLLA	90-degree delay for the DQS Input during a memory interface or the clock input for a generic DDR interface
ECLKSYNCB	ECLK Stop Block
IDDR71B	7:1 LVDS Input Supporting 1:7 Gearing
IDDRX1F	Generic Input DDR primitive
IDDRX2F	Generic Input DDR primitive
IDDRX4C	DDR primitive
IDDR141A	DDR primitive
IDDRX8A	DDR primitive
ODDRX1F	Generic X1 ODDR implementation

Table 490: Dual Data Rate Cells

ODDRX2F	Generic X2 ODDR implementation
ODDRX4C	ODDR primitive
ODDR141A	ODDR primitive
ODDRX8A	ODDR primitive
ODDR71B	7:1 LVDS ODDR implementation

Table 491: Miscellaneous

BCINRD	Dynamic Bank Controller InRD
BCLVDSOB	Bank Controller for LVDS Outut Buffers
CCU2C	Carry-Chain
DELAYF	Delay
DELAYG	Delay
GSR	Global Set/Reset
INRDB	Input Reference and Differential Buffer
I2CA	I2C Primitive
LVDSOB	LVDS Output Buffer
MIPI	Special Primitive for MIPI Input Support
MIPIDPHYA	Primitive
PFUMX	2-Input mux within the PFU, C0 used for Selection with Positive Select
PMUA	Power Management Unit
PUR	Power Up Set/Reset
SGSR	Synchronous Release Global Set/Reset Interface

Primitive Library - LIFMDF

This library includes compatible FPGA primitives supported by the LIFMDF device family.

- Flip-Flops
- Input/Output Buffers
- Memory Primitives
- Logic Gates
- Miscellaneous Logic
- Multiplexers
- PIC Cells
  - PIC Flip-Flops (Input)
  - PIC Flip-Flops (Output)
  - PIC Latches (Input)
- Read-Only Memory
- Special Cells
  - Clock Manager/PLL/DLL
  - Combinatorial Primitives
  - Miscellaneous

References

For further information on individual primitives, a variety of technical notes for the LIFMD family are available on the Lattice web site.

- TN1301 – CrossLink High-Speed I/O Interface
- TN1302 – CrossLink Hardware Checklist
- TN1303 – CrossLink Programming and Configuration Usage Guide
- TN1304 – CrossLink sysCLOCK PLL/DLL Design and Usage Guide
- TN1305 – CrossLink sysI/O Usage Guide
- TN1306 – CrossLink Memory Usage Guide
- TN1307 – Power Management and Calculation for CrossLink Devices
- TN1308 – CrossLink I2C Hardened IP Usage Guide
- TN1309 – Advanced CrossLink I2C Hardened IP Reference Guide

Table 492: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset

Table 492: Flip-Flops (Continued)

FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 493: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
IB	CMOS Input Buffer
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OLVDS	LVDS Output Buffer

Table 494: Memory Primitives

DP8KE	True Dual Port EBR RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
PDPW8KE	Pseudo Dual Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 495: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate

Table 495: Logic Gates

NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 496: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 497: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

PIC Cells

Table 498: PIC Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 499: PIC Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 500: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 501: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory

Table 501: Read-Only Memory

ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 502: Clock Manager/PLL/DLL

CLKDIVG	Clock Divider
DCCA	Dynamic Clock Control Block
DCSC	Dynamic Clock Selection
DLLDELD	Slave Delay
EHXPLLM	GPLL
OSCI	Oscillator for Configuration Clock
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 503: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 504: Dual Data Rate Cells

DDRDLLA	90-degree delay for the DQS Input during a memory interface or the clock input for a generic DDR interface
ECLKSYNCB	ECLK Stop Block
IDDR71B	7:1 LVDS Input Supporting 1:7 Gearing
IDDRX1F	Generic Input DDR primitive
IDDRX2F	Generic Input DDR primitive
IDDRX4C	DDR primitive
IDDR141A	DDR primitive
IDDRX8A	DDR primitive
ODDRX1F	Generic X1 ODDR implementation
ODDRX2F	Generic X2 ODDR implementation

Table 504: Dual Data Rate Cells

ODDRX4C	ODDR primitive
ODDR141A	ODDR primitive
ODDRX8A	ODDR primitive
ODDR71B	7:1 LVDS ODDR implementation

Table 505: Miscellaneous

BCINRD	Dynamic Bank Controller InRD
BCLVDSOB	Bank Controller for LVDS Outut Buffers
CCU2C	Carry-Chain
DELAYF	Delay
DELAYG	Delay
GSR	Global Set/Reset
INRDB	Input Reference and Differential Buffer
I2CA	I2C Primitive
LVDSOB	LVDS Output Buffer
MIPI	Special Primitive for MIPI Input Support
MIPIDPHYA	Primitive
PFUMX	2-Input mux within the PFU, C0 used for Selection with Positive Select
PMUA	Power Management Unit
PUR	Power Up Set/Reset
SGSR	Synchronous Release Global Set/Reset Interface

Primitive Library - MachXO and Platform Manager

This library includes compatible primitives supported by the MachXO and Platform Manager devices.

- Adders/Subtractors
- Comparators
- Counters
- Loadable Counters
- Flip-Flops
- Input/Output Buffers
- Latches
- Logic Gates
- MachXO and Platform Manager Memory Primitives
- Miselleaneous Logic
- Multiplexers
- Multipliers
- Read-Only Memory
- Combinatorial Primitives
- Miscellaneous

References

For further information, a variety of technical notes for the MachXO family are available on the Lattice Web site.

- TN1086 - MachXO JTAG Programming and Configuration User's Guide
- TN1087 - Minimizing System Interruption During Configuration Using TransFR Technology
- TN1089 - MachXO sysCLOCK PLL Design and Usage Guide
- TN1090 - Power Calculations and Considerations for MachXO Devices
- TN1091 - MachXO sysIO Usage Guide
- TN1092 - MachXO Memory Usage Guide
- TN1097 - MachXO Density Migration
- IEEE 1149.1 Boundary Scan Testability in Lattice Devices

Table 506: Adders/Subtractors

FADD2	2 Bit Fast Adder
FSUB2	2 Bit Fast Subtractor (two's complement)
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)

Table 507: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 508: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 509: Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD2P3AY	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)

Table 509: Loadable Counters (Continued)

LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)
LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 510: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear

Table 510: Flip-Flops (Continued)

FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 511: Input/Output Buffers

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer

Table 511: Input/Output Buffers

OB	Output Buffer
OBW	Output Buffer with Tristate
OBZ	Output Buffer with Tristate
OBZPD	Output Buffer with Tristate and Pull-down
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 512: Latches

FD1S1A	Positive Level Data Latch with GSR Used for Clear
FD1S1AY	Positive Level Data Latch with GSR Used for Preset
FD1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset
FD1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear
FD1S1I	Positive Level Data Latch with Positive Level Synchronous Clear
FD1S1J	Positive Level Data Latch with Positive Level Synchronous Preset
FL1S1A	Positive Level Loadable Latch with Positive Select and GSR Used for Clear
FL1S1AY	Positive Level Loadable Latch with Positive Select and GSR Used for Preset
FL1S1B	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Preset
FL1S1D	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Clear
FL1S1I	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Clear
FL1S1J	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Preset

Table 513: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate

Table 513: Logic Gates (Continued)

ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR2	2 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate
XOR11	11 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate

Table 514: MachXO and Platform Manager Memory Primitives

DP8KB	8K Dual Port Block RAM
DPR16X2B	16 Word by 2 Dual Port RAM (within PFU)
FIFO8KA	8K FIFO
PDP8KB	8K Pseudo Dual Port Block RAM
SP8KB	8 Word by 8 Bit Single Port Block RAM
SPR16X2B	16 Word by 2 Bit Positive Edge Triggered Write Synchronous Single Port RAM Memory with Positive Write Enable and Positive Write Port Enable (1-Slice)

Table 515: Miselleaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 516: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 517: Multipliers

MULT2	2x2 Multiplier

Table 518: Read-Only Memory

ROM16X1	16 Word by 1 bit read-only memory
ROM32X1	32 Word by 1 bit read-only memory
ROM64X1	64 Word by 1 bit read-only memory
ROM128X1	128 Word by 1 bit read-only memory
ROM256X1	256 Word by 1 bit read-only memory

Special Cells

Table 519: Combinatorial Primitives

ORCALUT4	4-Input Look Up Table
ORCALUT5	5-Input Look Up Table
ORCALUT6	6-Input Look Up Table
ORCALUT7	7-Input Look Up Table
ORCALUT8	8-Input Look Up Table

Table 520: Miscellaneous

CCU2	Carry Chain
EHXPLLC	Enhanced Extended Performance PLL
GSR	Global Set/Reset
JTAGD	JTAG (Joint Test Action Group) Controller
OSCC	Internal Oscillator
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PUR	Power Up Set/Reset
TSALL	Global Tristate Interface

Primitive Library - MachXO2 and Platform Manager 2

This library includes compatible primitives supported by the MachXO2 device family.

- Adders/Subtractors
- Comparators
- Counters
  - Bi-Directional Loadable Counters
  - Loadable Down Counters
  - Loadable Up Counters
- Flip-Flops
- Input/Output Buffer
- Latches
- Logic Gates
- PIC Cells
  - Flip-Flops (Input)
  - Flip-Flops (Output)
  - PIC Latches (Input)
- MachXO2/Platform Manager 2 Memory Primitives
- Miscellaneous Logic
- Multiplexers
- Multipliers
- Read-Only Memory
- Special Cells
  - Clock/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information, a variety of technical notes for the MachXO2 family are available on the Lattice Web site.

- TN1198 - Power Estimation and Management for MachXO2 Device
- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide
- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1202 - MachXO2 sysIO Usage Guide
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

- TN1204 - MachXO2 Programming and Configuration Usage Guide
- TN1205 - MachXO2 User Flash Memory and Hardened Control Functions
- TN1206 - MachXO2 Soft Error Detection (SED) Usage Guide

Table 521: Adders/Subtractors

FADD2B	Fast 2 Bit Adder
FSUB2B	2 Bit Subtractor
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)

Table 522: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 523: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 524: Bi-Directional Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 525: Loadable Down Counters

LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD2P3AY	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 526: Loadable Up Counters

LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 527: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset

Table 527: Flip-Flops (Continued)

FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 528: Input/Output Buffer

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 529: Latches

FD1S1A	Positive Level Data Latch with GSR Used for Clear
FD1S1AY	Positive Level Data Latch with GSR Used for Preset
FD1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset
FD1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear
FD1S1I	Positive Level Data Latch with Positive Level Synchronous Clear
FD1S1J	Positive Level Data Latch with Positive Level Synchronous Preset
FL1S1A	Positive Level Loadable Latch with Positive Select and GSR Used for Clear
FL1S1AY	Positive Level Loadable Latch with Positive Select and GSR Used for Preset
FL1S1B	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Preset

Table 529: Latches (Continued)

FL1S1D	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Clear
FL1S1I	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Clear
FL1S1J	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Preset

Table 530: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR11	11 Input Exclusive OR Gate
XOR2	2 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate

Table 530: Logic Gates (Continued)

XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate

PIC Cells

Table 531: Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 532: Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 533: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)

Table 533: PIC Latches (Input) (Continued)

IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 534: MachXO2/Platform Manager 2 Memory Primitives

DP8KC	8K True Dual Port Block RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
FIFO8KB	8K FIFO Block RAM
PDPW8KC	Pseudo Dual Port Block RAM
SP8KC	8K Single Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 535: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 536: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 537: Multipliers

MULT2	2x2 Multiplier

Table 538: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory
ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 539: Clock/PLL/DLL

CLKDIVC	Clock Divider
DCCA	Dynamic Quadrant Clock Enable/Disable
DCMA	Dynamic Clock Mux
DLLDELC	Clock Shifting for ECLK or PCLK
ECLKBRIDGECS	ECLK Bridge Block Clock Select
ECLKSYNCA	ECLK Stop Block
EHXPLLJ	GPLL for MachXO2
OSCH	Oscillator for MachXO2
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 540: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 541: Dual Data Rate Cells

DQSBUFH	DQS Circuit for DDR Memory
DQSDLLC	Master DLL for Generating Required Delay
IDDRXE	Input for Generic DDR X1 Using 1:2 Gearing
IDDRX2E	Input for Generic DDR X2 Using 1:4 Gearing
IDDRX4B	Input for Generic DDR X4 Using 1:8 Gearing
IDDRDQSX1A	Input for DDR1/2 Memory
IDDRX71A	7:1 LVDS Input Supporting 1:7 Gearing
ODDRXE	Output for Generic DDR X1 Using 2:1 Gearing
ODDRX2E	Output for Generic DDR X2 Using 4:1 Gearing
ODDRX4B	Output for Generic DDR X4 Using 8:1 Gearing
ODDRDQSX1A	Output for DDR1/2 Memory
ODDRX71A	7:1 LVDS Output
TDDRA	Tristate for DQ/DQS of PIC Cell

Table 542: Miscellaneous

BCINRD	Dynamic Bank Controller InRD
BCLVDSO	Dynamic Bank Controller LVDS
CCU2D	Carry Chain
CLKFBBUFA	Dummy Feedback Delay Between PLL clk Output and PLL fb Port
DELAYD	Dynamic Delay for Bottom Bank
DELAYE	Fixed Delay in PIO
EFB	Embedded Function Block
GSR	Global Set/Reset
INRDB	Input Reference and Differential Buffer
JTAGF	JTAG (Joint Test Action Group) Controller
LVDSOB	LVDS Output Buffer
PCNTR	Power Controller
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PG	Power Guard

Table 542: Miscellaneous (Continued)

PUR	Power Up Set/Reset
SEDFA	Soft Error Detect in Basic Mode
SEDFB	Soft Error Detect in One Shot Mode
SGSR	Synchronous Release Global Set/Reset Interface
START	Startup Controller
TSALL	Global Tristate Interface

Primitive Library - MachXO3D

This library includes compatible primitives supported by the MachXO3D device family.

- Adders/Subtractors
- Comparators
- Counters
  - Bi-Directional Loadable Counters
  - Loadable Down Counters
  - Loadable Up Counters
- Flip-Flops
- Input/Output Buffer
- Latches
- Logic Gates
- PIC Cells
  - Flip-Flops (Input)
  - Flip-Flops (Output)
  - PIC Latches (Input)
- Memory Primitives
- Miscellaneous Logic
- Multiplexers
- Multipliers
- Read-Only Memory
- Special Cells
  - Clock/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information, a variety of technical notes for the MachXO3L family are available on the Lattice web site.

- TN1281 - Implementing High-Speed Interfaces with MachXO3L Devices
- TN1291 - MachXO3L Hardware Checklist
- TN1279 - MachXO3L Programming and Configuration Usage Guide
- TN1292 - MachXO3L SED Usage Guide
- TN1282 - MachXO3L sysCLOCK PLL Design and Usage Guide
- TN1280 - MachXO3L sysIO Usage Guide

- TN1290 - Memory Usage Guide for MachXO3L Devices
- TN1293 - Using Hardened Control Functions in MachXO3L Devices
- TN1294 - Using Hardened Control Functions in MachXO3L Devices Reference Guide

Table 543: Adders/Subtractors

FADD2B	Fast 2 Bit Adder
FSUB2B	2 Bit Subtractor
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)

Table 544: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 545: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 546: Bi-Directional Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LB2P3DX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 547: Loadable Down Counters

LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD2P3AY	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 548: Loadable Up Counters

LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 549: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)

Table 549: Flip-Flops (Continued)

FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)
FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 550: Input/Output Buffer

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBI3C	I3C Bidirectional Buffer
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down

Table 550: Input/Output Buffer

BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 551: Latches

FD1S1A	Positive Level Data Latch with GSR Used for Clear
FD1S1AY	Positive Level Data Latch with GSR Used for Preset
FD1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset
FD1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear
FD1S1I	Positive Level Data Latch with Positive Level Synchronous Clear
FD1S1J	Positive Level Data Latch with Positive Level Synchronous Preset
FL1S1A	Positive Level Loadable Latch with Positive Select and GSR Used for Clear
FL1S1AY	Positive Level Loadable Latch with Positive Select and GSR Used for Preset
FL1S1B	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Preset
FL1S1D	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Clear
FL1S1I	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Clear
FL1S1J	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Preset

Table 552: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate

Table 552: Logic Gates (Continued)

AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate
XOR11	11 Input Exclusive OR Gate
XOR2	2 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate

PIC Cells

Table 553: Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 554: Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 555: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 556: Memory Primitives

DP8KC	8K True Dual Port Block RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
FIFO8KB	8K FIFO Block RAM

Table 556: Memory Primitives

PDPW8KC	Pseudo Dual Port Block RAM
SP8KC	8K Single Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 557: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 558: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 559: Multipliers

MULT2	2x2 Multiplier

Table 560: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory
ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 561: Clock/PLL/DLL

CLKDIVC	Clock Divider
DCCA	Dynamic Quadrant Clock Enable/Disable
DCMA	Dynamic Clock Mux

Table 561: Clock/PLL/DLL

DLLDELC	Clock Shifting for ECLK or PCLK
ECLKBRIDGECS	ECLK Bridge Block Clock Select
ECLKSYNCA	ECLK Stop Block
EHXPLLJ	GPLL
OSCJ	Oscillator for MachXO3D
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 562: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 563: Dual Data Rate Cells

DQSDLLC	Master DLL for Generating Required Delay
IDDRXE	Input for Generic DDR X1 Using 1:2 Gearing
IDDRX2E	Input for Generic DDR X2 Using 1:4 Gearing
IDDRX4B	Input for Generic DDR X4 Using 1:8 Gearing
IDDRX71A	7:1 LVDS Input Supporting 1:7 Gearing
ODDRXE	Output for Generic DDR X1 Using 2:1 Gearing
ODDRX2E	Output for Generic DDR X2 Using 4:1 Gearing
ODDRX4B	Output for Generic DDR X4 Using 8:1 Gearing
ODDRX71A	7:1 LVDS Output

Table 564: Miscellaneous

BCINRD	Dynamic Bank Controller InRD
BCLVDSO	Dynamic Bank Controller LVDS
BCSLEWRATEA	Bank Controller for Slew Rate
CCU2D	Carry Chain
CLKFBBUFA	Dummy Feedback Delay Between PLL clk Output and PLL fb Port
DELAYD	Dynamic Delay for Bottom Bank
DELAYE	Fixed Delay in PIO

Table 564: Miscellaneous (Continued)

EFBB	Embedded Function Block for MachXO3D
ESBA	Embedded Security Block for MachXO3D
GSR	Global Set/Reset
INRDB	Input Reference and Differential Buffer
JTAGF	JTAG (Joint Test Action Group) Controller
LVDSOB	LVDS Output Buffer
PCNTR	Power Controller
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PG	Power Guard
PUR	Power Up Set/Reset
SEDFA	Soft Error Detect in Basic Mode
SEDFB	Soft Error Detect in One Shot Mode
SGSR	Synchronous Release Global Set/Reset Interface
START	Startup Controller
TSALL	Global Tristate Interface

Primitive Library - MachXO3L

This library includes compatible primitives supported by the MachXO3L device family.

- Adders/Subtractors
- Comparators
- Counters
  - Bi-Directional Loadable Counters
  - Loadable Down Counters
  - Loadable Up Counters
- Flip-Flops
- Input/Output Buffer
- Latches
- Logic Gates
- PIC Cells
  - Flip-Flops (Input)
  - Flip-Flops (Output)
  - PIC Latches (Input)
- MachXO2 Memory Primitives
- Miscellaneous Logic
- Multiplexers
- Multipliers
- Read-Only Memory
- Special Cells
  - Clock/PLL/DLL
  - Combinatorial Primitives
  - Dual Data Rate Cells
  - Miscellaneous

References

For further information, a variety of technical notes for the MachXO3l family are available on the Lattice Web site.

- TN1281 - Implementing High-Speed Interfaces with MachXO3L Devices
- TN1291 - MachXO3L Hardware Checklist
- TN1279 - MachXO3L Programming and Configuration Usage Guide
- TN1292 - MachXO3L SED Usage Guide
- TN1282 - MachXO3L sysCLOCK PLL Design and Usage Guide
- TN1280 - MachXO3L sysIO Usage Guide

- TN1290 - Memory Usage Guide for MachXO3L Devices
- TN1293 - Using Hardened Control Functions in MachXO3L Devices
- TN1294 - Using Hardened Control Functions in MachXO3L Devices Reference Guide

Table 565: Adders/Subtractors

FADD2B	Fast 2 Bit Adder
FSUB2B	2 Bit Subtractor
FADSU2	2 Bit Fast Adder/Subtractor (two's complement)

Table 566: Comparators

AGEB2	A Greater Than Or Equal To B (2 bit)
ALEB2	A Less Than Or Equal To B (2 bit)
ANEB2	A Not Equal To B (2 bit)

Table 567: Counters

CB2	Combinational Logic for 2-Bit Bidirectional Counter
CD2	Combinational Logic for 2 Bit Down Counter
CU2	Combinational Logic for 2 Bit Up Counter

Table 568: Bi-Directional Loadable Counters

LB2P3AX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear
LB2P3AY	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset
LB2P3BX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset
	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear

Table 568: Bi-Directional Loadable Counters (Continued)

LB2P3IX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LB2P3JX	2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 569: Loadable Down Counters

LD2P3AX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable, GSR Used for Clear
LD2P3AY	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Preset
LD2P3BX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LD2P3DX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LD2P3IX	2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LD2P3JX	2 Bit Positive Edge Triggered Loadable Down Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 570: Loadable Up Counters

LU2P3AX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Clear
LU2P3AY	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable, GSR Used for Preset
LU2P3BX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Preset
LU2P3DX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Asynchronous Clear
LU2P3IX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)
LU2P3JX	2 Bit Positive Edge Triggered Loadable Up Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Table 571: Flip-Flops

FD1P3AX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Clear
FD1P3AY	Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR used for Preset
FD1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
FD1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
FD1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
FD1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
FD1S3AX	Positive Edge Triggered D Flip-Flop, GSR Used for Clear
FD1S3AY	Positive Edge Triggered D Flip-Flop, GSR Used for Preset
FD1S3BX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset
FD1S3DX	Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear
FD1S3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear
FD1S3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset
FL1P3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Preset
FL1P3AZ	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR used for Clear
FL1P3BX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable
FL1P3DX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable
FL1P3IY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)
FL1P3JY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)

Table 571: Flip-Flops (Continued)

FL1S3AX	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Clear
FL1S3AY	Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR used for Preset

Table 572: Input/Output Buffer

BB	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional
BBPD	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-down
BBPU	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional and Pull-up
BBW	CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate BiDirectional in keepermode
IB	Input Buffer
IBPD	Input Buffer with Pull-down
IBPU	Input Buffer with Pull-up
ILVDS	LVDS Input Buffer
OB	Output Buffer
OBCO	Output Complementary Buffer
OBZ	Output Buffer with Tristate
OBZPU	Output Buffer with Tristate and Pull-up
OLVDS	LVDS Output Buffer

Table 573: Latches

FD1S1A	Positive Level Data Latch with GSR Used for Clear
FD1S1AY	Positive Level Data Latch with GSR Used for Preset
FD1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset
FD1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear
FD1S1I	Positive Level Data Latch with Positive Level Synchronous Clear
FD1S1J	Positive Level Data Latch with Positive Level Synchronous Preset

Table 573: Latches (Continued)

FL1S1A	Positive Level Loadable Latch with Positive Select and GSR Used for Clear
FL1S1AY	Positive Level Loadable Latch with Positive Select and GSR Used for Preset
FL1S1B	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Preset
FL1S1D	Positive Level Loadable Latch with Positive Select and Positive Level Asynchronous Clear
FL1S1I	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Clear
FL1S1J	Positive Level Loadable Latch with Positive Select and Positive Level Synchronous Preset

Table 574: Logic Gates

AND2	2 Input AND Gate
AND3	3 Input AND Gate
AND4	4 Input AND Gate
AND5	5 Input AND Gate
ND2	2 Input NAND Gate
ND3	3 Input NAND Gate
ND4	4 Input NAND Gate
ND5	5 Input NAND Gate
OR2	2 Input OR Gate
OR3	3 Input OR Gate
OR4	4 Input OR Gate
OR5	5 Input OR Gate
NR2	2 Input NOR Gate
NR3	3 Input NOR Gate
NR4	4 Input NOR Gate
NR5	5 Input NOR Gate
XNOR2	2 Input Exclusive NOR Gate
XNOR3	3 Input Exclusive NOR Gate
XNOR4	4 Input Exclusive NOR Gate
XNOR5	5 Input Exclusive NOR Gate

Table 574: Logic Gates (Continued)

XOR11	11 Input Exclusive OR Gate
XOR2	2 Input Exclusive OR Gate
XOR21	21 Input Exclusive OR Gate
XOR3	3 Input Exclusive OR Gate
XOR4	4 Input Exclusive OR Gate
XOR5	5 Input Exclusive OR Gate

PIC Cells

Table 575: Flip-Flops (Input)

IFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)
IFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)
IFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)
IFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Table 576: Flip-Flops (Output)

OFS1P3BX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)
OFS1P3DX	Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)
OFS1P3IX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)
OFS1P3JX	Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Table 577: PIC Latches (Input)

IFS1S1B	Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)
IFS1S1D	Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)
IFS1S1I	Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)
IFS1S1J	Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Table 578: MachXO2 Memory Primitives

DP8KC	8K True Dual Port Block RAM
DPR16X4C	Distributed Pseudo Dual Port RAM
FIFO8KB	8K FIFO Block RAM
PDPW8KC	Pseudo Dual Port Block RAM
SP8KC	8K Single Port Block RAM
SPR16X4C	Distributed Single Port RAM

Table 579: Miscellaneous Logic

INV	Inverter
VHI	Logic High Generator
VLO	Logic Low Generator

Table 580: Multiplexers

L6MUX21	LUT-6 2 to 1 Multiplexer
MUX161	16-Input Mux within the PFU (4 Slices)
MUX21	2 to 1 Mux
MUX321	32-Input Mux within the PFU (8 Slices)
MUX41	4 to 1 Mux
MUX81	8 to 1 Mux

Table 581: Multipliers

MULT2	2x2 Multiplier

Table 582: Read-Only Memory

ROM128X1A	128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM16X1A	16 Word by 1 Bit Read-Only Memory
ROM256X1A	256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs
ROM32X1A	32 Word by 1 Bit Read-Only Memory
ROM64X1A	64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Special Cells

Table 583: Clock/PLL/DLL

CLKDIVC	Clock Divider
DCCA	Dynamic Quadrant Clock Enable/Disable
DCMA	Dynamic Clock Mux
DLLDELC	Clock Shifting for ECLK or PCLK
ECLKBRIDGECS	ECLK Bridge Block Clock Select
ECLKSYNCA	ECLK Stop Block
EHXPLLJ	GPLL for MachXO2
OSCH	Oscillator for MachXO2
PLLREFCS	PLL Dynamic Reference Clock Switching

Table 584: Combinatorial Primitives

LUT4	4-Input Look Up Table
LUT5	5-Input Look Up Table
LUT6	6-Input Look Up Table
LUT7	7-Input Look Up Table
LUT8	8-Input Look Up Table

Table 585: Dual Data Rate Cells

DQSDLLC	Master DLL for Generating Required Delay
IDDRXE	Input for Generic DDR X1 Using 1:2 Gearing
IDDRX2E	Input for Generic DDR X2 Using 1:4 Gearing
IDDRX71A	7:1 LVDS Input Supporting 1:7 Gearing
ODDRXE	Output for Generic DDR X1 Using 2:1 Gearing
ODDRX2E	Output for Generic DDR X2 Using 4:1 Gearing
ODDRX71A	7:1 LVDS Output

Table 586: Miscellaneous

BCINRD	Dynamic Bank Controller InRD
BCLVDSO	Dynamic Bank Controller LVDS
CCU2D	Carry Chain
CLKFBBUFA	Dummy Feedback Delay Between PLL clk Output and PLL fb Port
DELAYD	Dynamic Delay for Bottom Bank
DELAYE	Fixed Delay in PIO
EFB	Embedded Function Block
GSR	Global Set/Reset
INRDB	Input Reference and Differential Buffer
JTAGF	JTAG (Joint Test Action Group) Controller
LVDSOB	LVDS Output Buffer
PCNTR	Power Controller
PFUMX	2-Input Mux within the PFU, C0 used for Selection with Positive Select
PG	Power Guard
PUR	Power Up Set/Reset
SEDFA	Soft Error Detect in Basic Mode
SEDFB	Soft Error Detect in One Shot Mode
SGSR	Synchronous Release Global Set/Reset Interface
START	Startup Controller
TSALL	Global Tristate Interface

Alphanumeric Primitives List

This section lists all the Lattice library primitives in alphanumeric order.

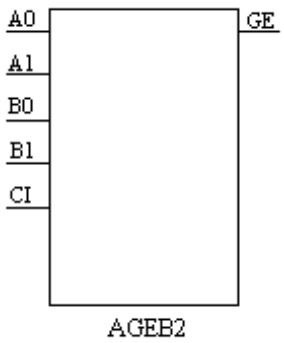
A

AGEB2

"A" Greater Than Or Equal To "B"

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A0, A1, B0, B1, CI

OUTPUT: GE

Description

AGEB2 is a 2-bit comparator that can be cascaded together to build larger comparators. It has two 2-bit inputs and a carry-in input. The carry-in (CI) on the first stage should be tied HIGH. The compare-out (GE) output is HIGH if A[1:0] >= B[1:0] and LOW if A[1:0] < B[1:0]. To build larger comparators, tie the GE on the lower stage to CI on the upper stage.

Note

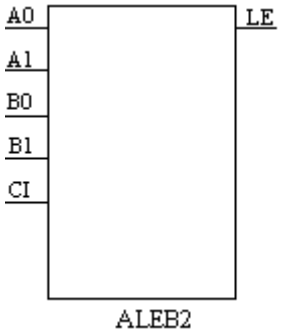
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ALEB2

"A" Less Than Or Equal To "B"

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A0, A1, B0, B1, CI

OUTPUT: LE

Description

ALEB2 is a 2-bit comparator that can be cascaded together to build larger comparators. It has two 2-bit inputs and a carry-in input. The carry-in (CI) on the first stage should be tied HIGH. The compare-out (LE) output is HIGH if A[1:0] <= B[1:0] and LOW if A[1:0] > B[1:0]. To build larger comparators, tie the LE on the lower stage to CI on the upper stage.

Note

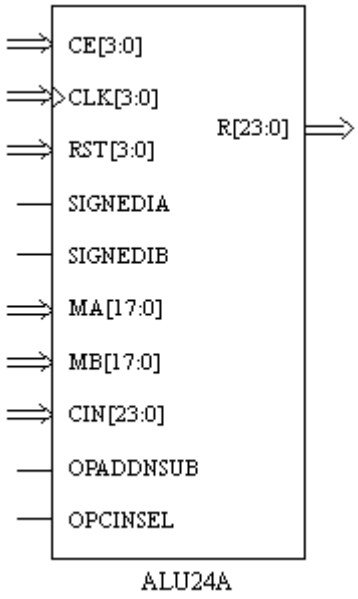
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ALU24A

24 Bit Ternary Adder/Subtractor

Architectures Supported:

- ECP5
- LatticeECP3



INPUTS: MA17, MA16, MA15, MA14, MA13, MA12, MA11, MA10, MA9, MA8, MA7, MA6, MA5, MA4, MA3, MA2, MA1, MA0, MB17, MB16, MB15, MB14, MB13, MB12, MB11, MB10, MB9, MB8, MB7, MB6, MB5, MB4, MB3, MB2, MB1, MB0, CIN23, CIN22, CIN21, CIN20, CIN19, CIN18, CIN17, CIN16, CIN15, CIN14, CIN13, CIN12, CIN11, CIN10, CIN9, CIN8, CIN7, CIN6, CIN5, CIN4, CIN3, CIN2, CIN1, CIN0, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SIGNEDIA, SIGNEDIB, OPADDNSUB, OPCINSEL

OUTPUTS: R23, R22, R21, R20, R19, R18, R17, R16, R15, R14, R13, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, R0

ATTRIBUTES:

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODE_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODE_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODE_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODE_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODE_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODE_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

ALU24A Attribute Description

Table 587:

Name	Description
REG_OUTPUT_CLK	ALU register for output clock selection
REG_OUTPUT_CLK	ALU register for output clock enable selection
REG_OUTPUT_RST	ALU register for output reset selection
REG_OPCODE_0_CLK	OPCODE register clock selection
REG_OPCODE_0_CE	OPCODE register clock enable selection
REG_OPCODE_0_RST	OPCODE register reset selection
REG_OPCODE_1_CLK	OPCODE pipeline register clock selection
REG_OPCODE_1_CE	OPCODE pipeline register clock enable selection
REG_OPCODE_1_RST	OPCODE pipeline register reset selection
GSR	Global set reset selection
RESETMODE	Reset mode selection

ALU24A Port Description

Table 588:

Input/Output	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	CLK0	CLK0	Bit	N/A	Clock Input
I	CLK1	CLK1	Bit	N/A	Clock Input
I	CLK2	CLK2	Bit	N/A	Clock Input
I	CLK3	CLK3	Bit	N/A	Clock Input
I	CE0	CE0	Bit	N/A	Clock Enable Input
I	CE1	CE1	Bit	N/A	Clock Enable Input
I	CE2	CE2	Bit	N/A	Clock Enable Input
I	CE3	CE3	Bit	N/A	Clock Enable Input
I	RST0	RST0	Bit	N/A	Reset Input
I	RST1	RST1	Bit	N/A	Reset Input
I	RST2	RST2	Bit	N/A	Reset Input

Table 588:

Input/Output	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	RST3	RST3	Bit	N/A	Reset Input
I	SIGNEDIA[1]	SIGNEDIA	Bit	N/A	Input A Sign Selection
I	SIGNEDIB[1]	SIGNEDIB	Bit	N/A	Input A Sign Selection
I	MA[1]	MA[35:18]	Bus	17:0	Input A from Multiplier
I	MB[1]	MB[35:18]	Bus	17:0	Input B from Multiplier
I	CIN	CIN[50:27]	Bus	23:0	CIN Input
I	OPCINSEL	OP5	Bit	N/A	CIN Select selects CIN (010) or GND (000)
I	OPADDNSUB	OP7	Bit	N/A	Add/Subtract selection
O	R	R[43:34]	Bus	23:0	Output

Notes:
1. A and B refer to the first and second multiplier of the slice and not the Ax and Bx inputs to multiplier x.

Note

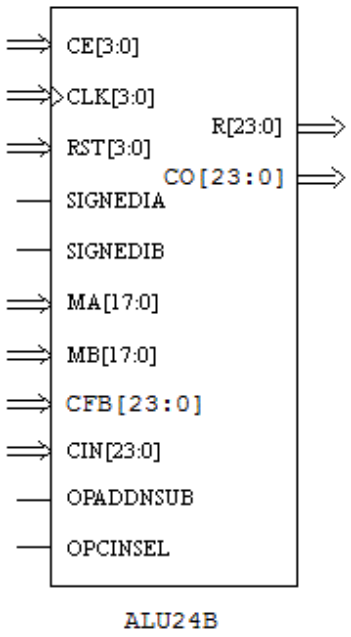
The synthesis tool will not use this block to inference DSP function.

ALU24B

24-bit Ternary Adder/Subtractor for 9x9 Mode

Architectures Supported:

- ECP5



INPUTS: MA17, MA16, MA15, MA14, MA13, MA12, MA11, MA10, MA9, MA8, MA7, MA6, MA5, MA4, MA3, MA2, MA1, MA0, MB17, MB16, MB15, MB14, MB13, MB12, MB11, MB10, MB9, MB8, MB7, MB6, MB5, MB4, MB3, MB2, MB1, MB0, CIN23, CIN22, CIN21, CIN20, CIN19, CIN18, CIN17, CIN16, CIN15, CIN14, CIN13, CIN12, CIN11, CIN10, CIN9, CIN8, CIN7, CIN6, CIN5, CIN4, CIN3, CIN2, CIN1, CIN0, CBF23, CBF22, CBF21, CBF20, CBF19, CBF18, CBF17, CBF16, CBF15, CBF14, CBF13, CBF12, CBF11, CBF10, CBF9, CBF8, CBF7, CBF6, CBF5, CBF4, CBF3, CBF2, CBF1, CBF0, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SIGNEDIA, SIGNEDIB, OPADDNSUB, OPCINSEL

OUTPUTS: R23, R22, R21, R20, R19, R18, R17, R16, R15, R14, R13, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, R0, CO23, CO22, CO21, CO20, CO19, CO18, CO17, CO16, CO15, CO14, CO13, CO12, CO11, CO10, CO9, CO8, CO7, CO6, CO5, CO4, CO3, CO2, CO1, CO0

ATTRIBUTES:

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODE_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODE_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODE_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODE_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODE_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODE_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTCFB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTCFB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTCFB_RST: "RST0" (default), "RST1", "RST2", "RST3"

CLK0_DIV: "ENABLED" (default), "DISABLED"

CLK1_DIV: "ENABLED" (default), "DISABLED"

CLK2_DIV: "ENABLED" (default), "DISABLED"

CLK3_DIV: "ENABLED" (default), "DISABLED"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

* When REG_INPUTCFB_CLK = NONE, then it means that the CFB ports are not used, and C -> Cr using the C0/C1_CLK attributes.

* When REG_INPUTCFB_CLK != NONE, then the CFB ports are being used and CFB -> CO is using these attributes. C -> Cr is unregistered and a DRC can check to make sure C0/C1_CLK = NONE

ALU24B Attribute Description

Table 589:

Attribute Name	Values	Default Value	GUI Access
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTCFB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTCFB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTCFB_RST	RST0, RST1, RST2, RST3	RST0	Y
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y
RESETMODE	SYNC, ASYNC	SYNC	Y
GSR	ENABLED, DISABLED	ENABLED	N

ALU24B Port Description

Table 590:

Port Name	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Input
RST[3:0]	I	Reset inputs
SIGNEDIA	I	Sign Bit for Input A

Table 590:

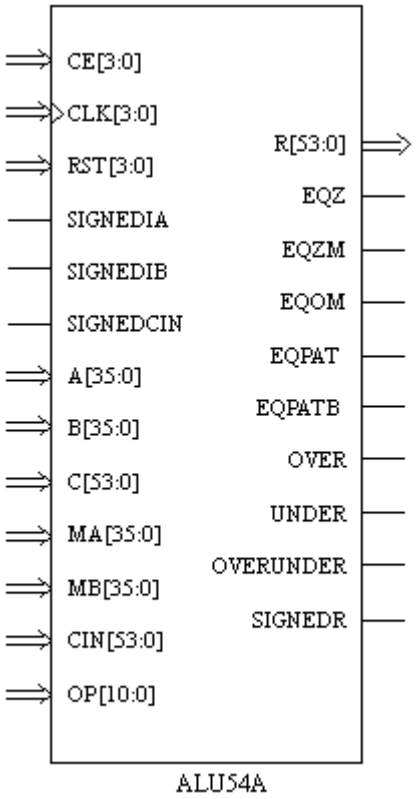
Port Name	I/O	Description
SIGNEDIB	I	Sign Bit for Input B
MA[17:0]	I	Input A
MB[17:0]	I	Input B
CFB[23:0]	I	C Input for Highspeed
CIN[23:0]	I	Carry In Input
OPADDNSUB	I	Add/Sub Selector
OPCINSEL	I	CarryIn Selector
R[23:0]	O	Sum
CO[23:0]	O	Sum – Special Routing output used for Highspeed option

ALU54A

54 Bit Ternary Adder/Subtractor

Architectures Supported:

- ECP5
- LatticeECP3



INPUTS: A35, A34, A33, A32, A31, A30, A29, A28, A27, A26, A25, A24, A23, A22, A21, A20, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B35, B34, B33, B32, B31, B30, B29, B28, B27, B26, B25, B24, B23, B22, B21, B20, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, C53, C52, C51, C50, C49, C48, C47, C46, C45, C44, C43, C42, C41, C40, C39, C38, C37, C36, C35, C34, C33, C32, C31, C30, C29, C28, C27, C26, C25, C24, C23, C22, C21, C20, C19, C18, C17, C16, C15, C14, C13, C12, C11, C10, C9, C8, C7, C6, C5, C4, C3, C2, C1, C0, MA35, MA34, MA33, MA32, MA31, MA30, MA29, MA28, MA27, MA26, MA25, MA24, MA23, MA22, MA21, MA20, MA19, MA18, MA17, MA16, MA15, MA14, MA13, MA12, MA11, MA10, MA9, MA8, MA7, MA6, MA5, MA4, MA3, MA2, MA1, MA0, MB35, MB34, MB33, MB32, MB31, MB30, MB29, MB28, MB27, MB26, MB25, MB24, MB23, MB22, MB21, MB20, MB19, MB18, MB17, MB16, MB15, MB14, MB13, MB12, MB11, MB10, MB9, MB8, MB7, MB6, MB5, MB4, MB3, MB2, MB1, MB0, CIN53, CIN52, CIN51, CIN50, CIN49, CIN48, CIN47, CIN46, CIN45, CIN44, CIN43, CIN42, CIN41, CIN40, CIN39, CIN38, CIN37, CIN36, CIN35, CIN34, CIN33, CIN32, CIN31, CIN30, CIN29, CIN28, CIN27, CIN26, CIN25, CIN24, CIN23, CIN22, CIN21, CIN20, CIN19, CIN18, CIN17, CIN16, CIN15, CIN14, CIN13, CIN12, CIN11, CIN10, CIN9, CIN8, CIN7, CIN6, CIN5, CIN4, CIN3, CIN2, CIN1, CIN0, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SIGNEDIA, SIGNEDIB, SIGNEDCIN, OP10, OP9, OP8, OP7, OP6, OP5, OP4, OP3, OP2, OP1, OP0

OUTPUTS: R53, R52, R51, R50, R49, R48, R47, R46, R45, R44, R43, R42, R41, R40, R39, R38, R37, R36, R35, R34, R33, R32, R31, R30, R29, R28, R27, R26, R25, R24, R23, R22, R21, R20, R19, R18, R17, R16, R15, R14, R13, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, R0, EQZ, EQZM, EQOM, EQPAT, EQPATB, OVER, UNDER, OVERUNDER, SIGNEDR

ATTRIBUTES:

REG_INPUTC0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTC0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTC0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTC1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTC1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTC1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEOP0_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEOP0_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEOP0_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEOP1_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEOP0_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEOP0_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEOP0_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEOP1_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEIN_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEIN_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEIN_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEIN_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEIN_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEIN_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_FLAG_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_FLAG_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_FLAG_RST: "RST0" (default), "RST1", "RST2", "RST3"

MCPAT_SOURCE: "STATIC" (default), "DYNAMIC"

MASKPAT_SOURCE: "STATIC" (default), "DYNAMIC"

MASK01: any 14-bit hexadecimal value (default: all zeros)

MCPAT: any 14-bit hexadecimal value (default: all zeros)

MASKPAT: any 14-bit hexadecimal value (default: all zeros)

RNDPAT: any 14-bit hexadecimal value (default: all zeros)

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

MULT9_MODE: "DISABLED" (default), "ENABLED"

FORCE_ZERO_BARREL_SHIFT: "DISABLED" (default), "ENABLED"

LEGACY: "DISABLED" (default), "ENABLED"

ALU54A Attribute Description

Table 591:

Name	Description
REG_INPUTC0_CLK	Input C register selection for C[26:0]
REG_INPUTC0_CE	Input C clock enable selection for C[26:0]
REG_INPUTC0_RST	Input C reset selection for C[26:0]
REG_INPUTC1_CLK	Input C register selection for C[53:27]
REG_INPUTC1_CE	Input C clock enable selection for C[53:27]
REG_INPUTC1_RST	Input C reset selection for C[53:27]
REG_OPCODEOP0_0_CLK	OPCODE register clock selection for oper [0]
REG_OPCODEOP0_0_CE	OPCODE register clock enable selection for oper [3:0]
REG_OPCODEOP0_0_RST	OPCODE register reset selection for oper [3:0]
REG_OPCODEOP1_0_CLK	OPCODE register clock selection for oper [3:1]
REG_OPCODEOP0_1_CLK	OPCODE pipeline register clock selection for oper [3:1]
REG_OPCODEOP0_1_CE	OPCODE pipeline register clock enable selection for oper [3:0]
REG_OPCODEOP0_1_RST	OPCODE pipeline register reset selection for oper [3:0]
REG_OPCODEOP1_1_CLK	OPCODE pipeline register clock selection for oper [3:1]
REG_OPCODEIN_0_CLK	OPCODE input register clock for InA[1:0], InB[1:0], InC[2:0]
REG_OPCODEIN_0_CE	OPCODE input register clock enable for InA[1:0], InB[1:0], InC[2:0]
REG_OPCODEIN_0_RST	OPCODE input register reset for InA[1:0], InB[1:0], InC[2:0]
REG_OPCODEIN_1_CLK	OPCODE input pipeline register clock for InA[1:0], InB[1:0], InC[2:0]
REG_OPCODEIN_1_CE	OPCODE input pipeline register clock enable for InA[1:0], InB[1:0], InC[2:0]
REG_OPCODEIN_1_RST	OPCODE input pipeline register reset for InA[1:0], InB[1:0], InC[2:0]
REG_OUTPUT0_CLK	ALU register for LSB output 17:0 clock selection
REG_OUTPUT0_CE	ALU register for LSB output 17:0 clock enable selection
REG_OUTPUT0_RST	ALU register for LSB output 17:0 reset selection
REG_OUTPUT1_CLK	ALU register for MSB output 53:18 clock selection

Table 591:

Name	Description
REG_OUTPUT1_CE	ALU register for MSB output 53:18 clock enable selection
REG_OUTPUT1_RST	ALU register for MSB output 53:18 reset selection
REG_FLAG_CLK	Flag register clock selection
REG_FLAG_CE	Flag register clock enable selection
REG_FLAG_RST	Flag pipeline register reset selection
MASKPAT_SOURCE[1]	EQPAT/EQPATB source setting
MCPAT_SOURCE[1]	MEM Cell Pattern source setting
MASK01	Mask for EQZM/EQOM
MASKPAT	Mask for EQPAT/EQPATB
MCPAT	MEM Cell Pattern
RNDPAT	Rounding Pattern
GSR	Global set reset selection
RESETMODE	Reset mode selection
MULT9_MODE	Operation in Mult9 mode
FORCE_ZERO_BARREL_SHIFT	Forces zeros to 18 MSB of shift for barrel shift
LEGACY	Required to support LatticeECP2 backward compatibility

Notes:
1. MASKPAT_SOURCE and MCPAT_SOURCE cannot be DYNAMIC at the same time since both use C[53:0]. There should be a DRC in the software to check this.

ALU54A Port Description

Table 592:

Input/Output	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	CLK0	CLK0	Bit	N/A	Clock Input
I	CLK1	CLK1	Bit	N/A	Clock Input
I	CLK2	CLK2	Bit	N/A	Clock Input
I	CLK3	CLK3	Bit	N/A	Clock Input
I	CE0	CE0	Bit	N/A	Clock Enable Input
I	CE1	CE1	Bit	N/A	Clock Enable Input
I	CE2	CE2	Bit	N/A	Clock Enable Input
I	CE3	CE3	Bit	N/A	Clock Enable Input

Table 592:

Input/Output	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	RST0	RST0	Bit	N/A	Reset Input
I	RST1	RST1	Bit	N/A	Reset Input
I	RST2	RST2	Bit	N/A	Reset Input
I	RST3	RST3	Bit	N/A	Reset Input
I	SIGNEDIA[1]	SIGNEDIA	Bit	N/A	Input A Sign Selection
I	SIGNEDIB[1]	SIGNEDIB	Bit	N/A	Input A Sign Selection
I	A[1]	A	Bus	35:0	Input A from Multiplier
I	B[1]	B	Bus	35:0	Input B from Multiplier
I	C	C	Bus	53:0	C Input
I	MA[1]	MA	Bus	35:0	Input A from Multiplier
I	MB[1]	MB	Bus	35:0	Input B from Multiplier
I	CIN	CIN	Bus	53:0	CIN Input
I	OP	OP	Bus	10:0	Opcode for ALU Operation Selection
I	SIGNEDCIN	SIGNEDCIN	Bit	N/A	CIN Right Shift, Signed or Unsigned Control
O	R	R	Bus	53:0	Output
O	EQZ	F[7]	Bit	N/A	Equal to Zero
O	EQZM	F[6]	Bit	N/A	Equal to Zero with Mask
O	EQPOM	F[5]	Bit	N/A	Equal to One with Mask
O	EQPAT	F[4]	Bit	N/A	Equal to Pat with Mask
O	EQPATB	F[3]	Bit	N/A	Equal to Bit Inverted Pat with Mask
O	OVER	F[2]	Bit	N/A	Accumulator Overflow
O	UNDER	F[1]	Bit	N/A	Accumulator Underflow
O	OVERUNDER	F[0]	Bit	N/A	Either Over or Under Flow
O	SIGNEDR	SIGNEDR	Bit	N/A	Signed or Unsigned Output of ALU

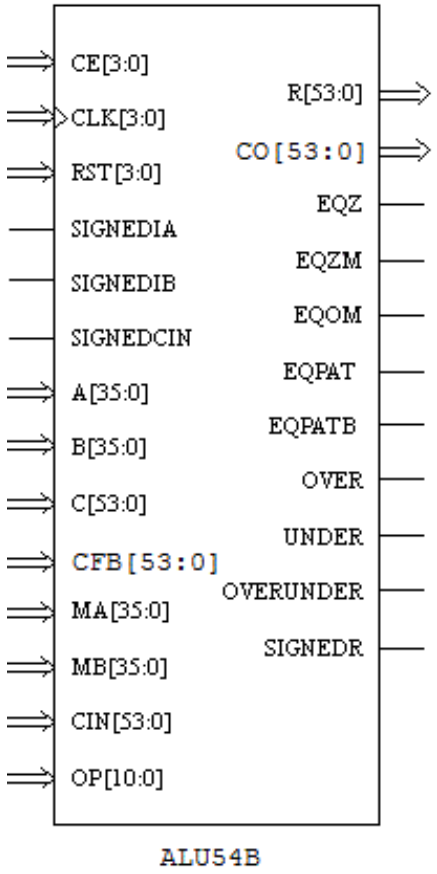
Notes:
1. A and B refer to the first and second multiplier of the slice and not the Ax and Bx inputs to multiplier x.

ALU54B

54-bit Ternary Adder/Subtractor for Highspeed

Architectures Supported:

- ECP5



INPUTS: A35, A34, A33, A32, A31, A30, A29, A28, A27, A26, A25, A24, A23, A22, A21, A20, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B35, B34, B33, B32, B31, B30, B29, B28, B27, B26, B25, B24, B23, B22, B21, B20, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, C53, C52, C51, C50, C49, C48, C47, C46, C45, C44, C43, C42, C41, C40, C39, C38, C37, C36, C35, C34, C33, C32, C31, C30, C29, C28, C27, C26, C25, C24, C23, C22, C21, C20, C19, C18, C17, C16, C15, C14, C13, C12, C11, C10, C9, C8, C7, C6, C5, C4, C3, C2, C1, C0, CFB53, CFB52, CFB51, CFB50, CFB49, CFB48, CFB47, CFB46, CFB45, CFB44, CFB43, CFB42, CFB41, CFB40, CFB39, CFB38, CFB37, CFB36, CFB35, CFB34, CFB33, CFB32, CFB31, CFB30, CFB29, CFB28, CFB27, CFB26, CFB25, CFB24, CFB23, CFB22, CFB21, CFB20, CFB19, CFB18, CFB17, CFB16, CFB15, CFB14, CFB13, CFB12, CFB11, CFB10, CFB9, CFB8, CFB7, CFB6, CFB5, CFB4, CFB3, CFB2, CFB1, CFB0, MA35, MA34, MA33, MA32, MA31, MA30, MA29, MA28, MA27, MA26, MA25, MA24, MA23, MA22, MA21, MA20, MA19, MA18,

MA17, MA16, MA15, MA14, MA13, MA12, MA11, MA10, MA9, MA8, MA7, MA6, MA5, MA4, MA3, MA2, MA1, MA0, MB35, MB34, MB33, MB32, MB31, MB30, MB29, MB28, MB27, MB26, MB25, MB24, MB23, MB22, MB21, MB20, MB19, MB18, MB17, MB16, MB15, MB14, MB13, MB12, MB11, MB10, MB9, MB8, MB7, MB6, MB5, MB4, MB3, MB2, MB1, MB0, CIN53, CIN52, CIN51, CIN50, CIN49, CIN48, CIN47, CIN46, CIN45, CIN44, CIN43, CIN42, CIN41, CIN40, CIN39, CIN38, CIN37, CIN36, CIN35, CIN34, CIN33, CIN32, CIN31, CIN30, CIN29, CIN28, CIN27, CIN26, CIN25, CIN24, CIN23, CIN22, CIN21, CIN20, CIN19, CIN18, CIN17, CIN16, CIN15, CIN14, CIN13, CIN12, CIN11, CIN10, CIN9, CIN8, CIN7, CIN6, CIN5, CIN4, CIN3, CIN2, CIN1, CIN0, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SIGNEDIA, SIGNEDIB, SIGNEDCIN, OP10, OP9, OP8, OP7, OP6, OP5, OP4, OP3, OP2, OP1, OP0

OUTPUTS: R53, R52, R51, R50, R49, R48, R47, R46, R45, R44, R43, R42, R41, R40, R39, R38, R37, R36, R35, R34, R33, R32, R31, R30, R29, R28, R27, R26, R25, R24, R23, R22, R21, R20, R19, R18, R17, R16, R15, R14, R13, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, R0, CO53, CO52, CO51, CO50, CO49, CO48, CO47, CO46, CO45, CO44, CO43, CO42, CO41, CO40, CO39, CO38, CO37, CO36, CO35, CO34, CO33, CO32, CO31, CO30, CO29, CO28, CO27, CO26, CO25, CO24, CO23, CO22, CO21, CO20, CO19, CO18, CO17, CO16, CO15, CO14, CO13, CO12, CO11, CO10, CO9, CO8, CO7, CO6, CO5, CO4, CO3, CO2, CO1, CO0, EQZ, EQZM, EQOM, EQPAT, EQPATB, OVER, UNDER, OVERUNDER, SIGNEDR

ATTRIBUTES:

REG_INPUTC0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTC0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTC0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTC1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTC1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTC1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEOP0_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEOP0_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEOP0_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEOP1_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEOP0_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEOP0_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEOP0_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEOP1_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEIN_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEIN_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEIN_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OPCODEIN_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OPCODEIN_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OPCODEIN_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_FLAG_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_FLAG_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_FLAG_RST: "RST0" (default), "RST1", "RST2", "RST3"

MCPAT_SOURCE: "STATIC" (default), "DYNAMIC"

MASKPAT_SOURCE: "STATIC" (default), "DYNAMIC"

MASK01: any 14-bit hexadecimal value (default: all zeros)

MCPAT: any 14-bit hexadecimal value (default: all zeros)

MASKPAT: any 14-bit hexadecimal value (default: all zeros)

RNDPAT: any 14-bit hexadecimal value (default: all zeros)

REG_INPUTCFB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTCFB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTCFB_RST: "RST0" (default), "RST1", "RST2", "RST3"

CLK0_DIV: "ENABLED" (default), "DISABLED"

CLK1_DIV: "ENABLED" (default), "DISABLED"

CLK2_DIV: "ENABLED" (default), "DISABLED"

CLK3_DIV: "ENABLED" (default), "DISABLED"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

MULT9_MODE: "DISABLED" (default), "ENABLED"

FORCE_ZERO_BARREL_SHIFT: "DISABLED" (default), "ENABLED"

LEGACY: "DISABLED" (default), "ENABLED"

Notes:

*REG_INPUT_C0 corresponds to the lower 27 bits of the C Input.

*REG_INPUT_C1 corresponds to the upper 27 bits of the C Input.

REG_OUTPUT0 corresponds to [17:0] of R and REG_OUTPUT1_* corresponds to [53:18] of R.

*When REG_INPUTCFB_CLK = NONE, then it means that the CFB ports are not used, and C -> Cr using the C0/C1_CLK attributes.

*When REG_INPUTCFB_CLK != NONE, then the CFB ports are being used and CFB -> CO is using these attributes. C -> Cr is unregistered and a DRC can check to make sure C0/C1_CLK = NONE.

ALU54B Attribute Description

Table 593:

Attribute Name	Values	Default Value	GUI Access
REG_INPUTC0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTC1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP0_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEIN_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y

Table 593:

Attribute Name	Values	Default Value	GUI Access
REG_OUTPUT0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_FLAG_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_FLAG_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_FLAG_RST	RST0, RST1, RST2, RST3	RST0	Y
MCPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASKPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASK01	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y
REG_INPUTCFB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTCFB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTCFB_RST	RST0, RST1, RST2, RST3	RST0	Y
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y
MCPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y
MASKPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y
RNDPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y
GSR	ENABLED, DISABLED	ENABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y
MULT9_MODE	ENABLED, DISABLED	DISABLED	N

Table 593:

Attribute Name	Values	Default Value	GUI Access
FORCE_ZERO_BARREL_SHIFT	ENABLED, DISABLED	DISABLED	N
LEGACY	ENABLED, DISABLED	DISABLED	Y

ALU24B Port Description

Table 594:

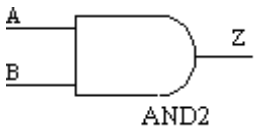
Port Name	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Input
RST[3:0]	I	Reset inputs
SIGNEDIA	I	Sign Bit for Input A
SIGNEDIB	I	Sign Bit for Input B
SIGNEDCIN	I	Sign Bit for Carrin In Input
A	I	Input A
B	I	Input B
C	I	Carry In Input/Highspeed Input
CFB[23:0]	I	C Input for Highspeed
MA[17:0]	I	Input A
MB[17:0]	I	Input B
CIN[23:0]	I	Carry In Input
OP[10:0]	I	Opcode
R[53:0]	O	Sum
CO[53:0]	O	Sum – Special Routing output used for Highspeed option
EQZ	O	Equal to Zero Flag
EQZM	O	Equal to Zero with Mask Flag
EQOM	O	Equal to One with Mask Flag
EQPAT	O	Equal to Pattern with Mask Flag
EQPATB	O	Equal to Bit Inverted Pattern with Mask Flag
OVER	O	Accumulator Overflow
UNDER	O	Accumulator Underflow
OVERUNDER	O	Either Over on Underflow
SIGNEDR	O	Sign Bit for Sum Output

AND2

2 Input AND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

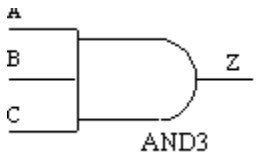
AND3

3 Input AND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3

- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

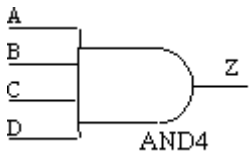
AND4

4 Input AND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX

- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

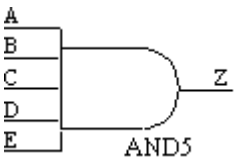
AND5

5 Input AND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E

OUTPUT: Z

Note

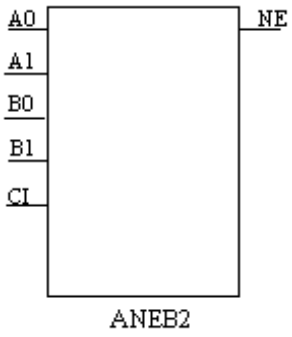
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ANEB2

"A" Not Equal To "B"

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A0, A1, B0, B1, CI

OUTPUT: NE

Description

ANEB2 is a 2-bit comparator that can be cascaded together to build larger comparators. It has two 2-bit inputs and a carry-in input. The carry-in (CI) on the first stage should be tied LOW. The compare-out (NE) output is LOW if A[1:0] = B[1:0] and HIGH otherwise. To build larger comparators, tie the NE on the lower stage to CI on the upper stage.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

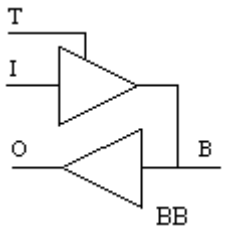
B

BB

CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate – BiDirectional

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: I, T

OUTPUT: O

IOPUT: B

Truth Table

Table 595:

INPUTS		OUTPUTS	BIDIRECTIONAL
I	T	O	B
X	1	U	Z
X	1	1	1
X	1	0	0
0	0	0	0
1	0	1	1

X = Don’t care

U = Unknown

When TSALL=0, O=U, B=Z

For PU/PD buffers, when TSALL=0, O and B will be pulled up or pulled down, respectively.

Note

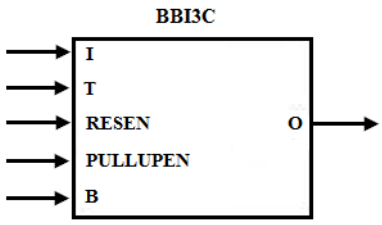
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

BBI3C

IC3 Bidirectional Buffer

Architectures Supported:

- MachXO3D



INPUTS: I, T, RESEN, PULLUPEN

OUTPUT: O

INOUT: B

Description

The following are descriptions of BBI3C port functions.

Table 596:

Port	I/O	Function
I	I	Input port
T	I	Tri-state control port
RESEN	I	Enables I3C strong pull-up (active low)
PULLUPEN	I	Enables I3C weak pull-up (active low)
B	I	Bidirectional port
O	O	Output port

Truth Table

Table 597:

INPUTS		OUTPUTS	BIDIRECTIONAL
I	T	O	B
X	1	U	Z
X	1	1	1
X	1	0	0
0	0	0	0
1	0	1	1

X = Don’t care

U = Unknown

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

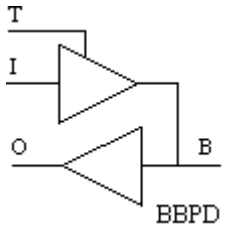
BBPD

CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate and Pull-down – BiDirectional

Architectures Supported:

- ECP5
- LatticeECP/EC

- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: I, T

OUTPUT: O

IOPUT: B

Truth Table

Table 598:

INPUTS		OUTPUTS	BIDIRECTIONAL
I	T	O	B
X	1	U	Pull0
X	1	1	1
X	1	0	0
0	0	0	0
1	0	1	1

X = Don’t care

U = Unknown

When TSALL=0, O=U, B=Z

For PU/PD buffers, when TSALL=0, O and B will be pulled up or pulled down, respectively.

Note

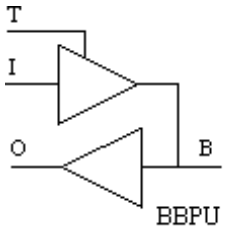
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

BBPU

CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate and Pull-up – BiDirectional

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: I, T

OUTPUT: O

IOPUT: B

Truth Table

Table 599:

INPUTS		OUTPUTS	BIDIRECTIONAL
I	T	O	B
X	1	U	Pull1
X	1	1	1
X	1	0	0
0	0	0	0
1	0	1	1

X = Don’t care

U = Unknown

When TSALL=0, O=U, B=Z

For PU/PD buffers, when TSALL=0, O and B will be pulled up or pulled down, respectively.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

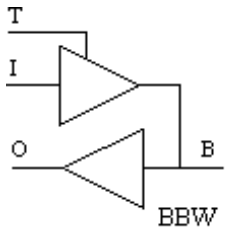
BBW

CMOS Input 6mA Sink 3mA Source Sinklim Output Buffer with Tristate – BiDirectional in keepermode

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: I, T

OUTPUT: O

IOPUT: B

Truth Table

Table 600:

INPUTS		OUTPUTS	BIDIRECTIONAL
I	T	O	B
0	1	0	weak 0
1	1	1	weak 1

X = Don’t care

U = Unknown

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

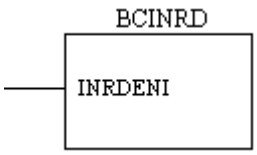
BCINRD

Dynamic Bank Controller InRD

Architectures Supported:

- ECP5
- LFMNX

- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUT: INRDENI

ATTRIBUTES:

BANKID: 0 (default), 1, 2, 3, 4, 5

Description

The dynamic bank controller is used to power down banks InRD (Input Referenced and Differential) and LVDS outputs. The dynamic bank controller is represented with two primitives: BCINRD and BCLVDSO. The INRDENI input is the dynamic signal to enable and disable bank InRD.

For more information, refer to the following technical note on the Lattice web site:

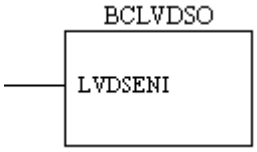
- TN1198 - Power Estimation and Management for MachXO2 Devices

BCLVDSO

Dynamic Bank Controller LVDS

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUT: LVDSENI

Description

The dynamic bank controller is used to power down banks InRD (Input Referenced and Differential) and LVDS outputs. The dynamic bank controller is represented with two primitives: BCLVDSO and BCINRD. The LVDSENI input is the dynamic signal to enable and disable bank InRD.

For more information, refer to the following technical note on the Lattice web site:

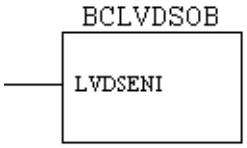
- TN1198 - Power Estimation and Management for MachXO2 Devices

BCLVDSOB

Bank Controller for LVDS Outut Buffers

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUT: LVDSENI

ATTRIBUTES:

BANKID:

ECP5: 2 (default), 2, 3, 6, 7

LIFMD/F: 1, 2

Description

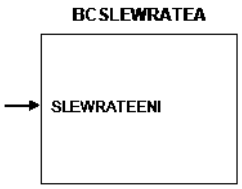
The dynamic bank controller is used to power down banks InRD (Input Referenced and Differential) and LVDS outputs. The LVDSENI input is the dynamic signal to enable and disable LVDS outputs.

BCSLEWRATEA

Bank Controller for Slew Rate

Architectures Supported:

- LFMNX
- MachXO3D



INPUT: SLEWRATEENI

Attributes:

Table 601: BCSLEWRATEA Primitives

Primitive Port	Primitive Attribute	Value	Description
SLEWRATEENI			Dynamic signal to enable and disable Bank Slewrate
	BANKID	0 (default), 1, 2, 3, 4, 5	Bank ID to indicate the control of a specific bank

Description

The dynamic bank controller is used to control the power down banks for slew rate slow mode. The SLEWRATEENI input is the dynamic signal to enable and disable the slew rate power saving per bank.

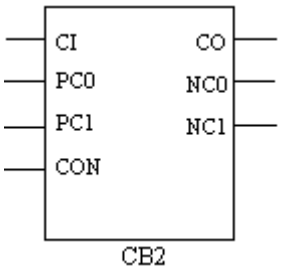
C

CB2

Combinational Logic for 2-Bit Bidirectional Counter

Architectures Supported:

- LatticeECP3
- LatticeXP2
- LatticeSC/M
- LatticeECP2/M
- LatticeECP/EC
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: CI, PC0, PC1, CON

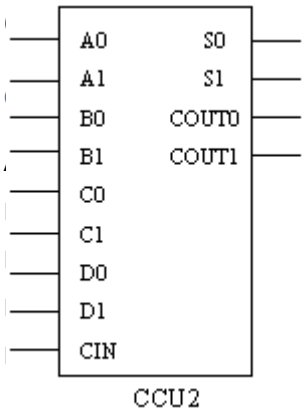
OUTPUTS: CO, NC0, NC1

Description

This primitive realizes the combinational logic needed to implement a 2-bit bidirectional counter by using ripple elements.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.



INPUT: CIN, A0, B0, C0, D0, A1, B1, C1, D1

OUTPUT: S0, S1, COUT0, COUT1

ATTRIBUTES:

INIT0: hexadecimal value (default: 16'h0000)

INIT1: hexadecimal value (default: 16'h0000)

INJECT1_0: "YES" (default), "NO"

INJECT1_1: "YES" (default), "NO"

Note

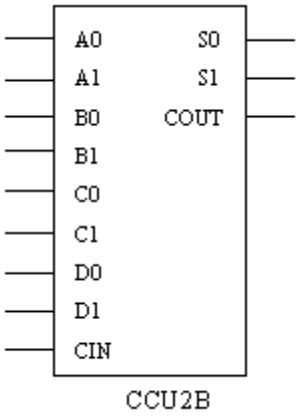
The attributes need to be defined when CCU2 is instantiated.

CCU2B

Carry-Chain

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: A0, B0, C0, D0, A1, B1, C1, D1, CIN

OUTPUTS: S0, S1, COUT

ATTRIBUTES:

INIT0: hexadecimal value (default: 16'h0000)

INIT1: hexadecimal value (default: 16'h0000)

INJECT1_0: "YES" (default), "NO"

INJECT1_1: "YES" (default), "NO"

Note

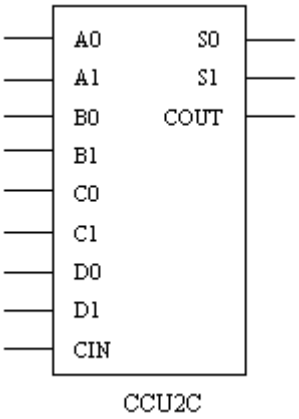
- The attributes need to be defined when CCU2B is instantiated.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

CCU2C

Carry Chain

Architectures Supported:

- ECP5
- LatticeECP3
- LIFMD
- LIFMDF



INPUTS: CIN, A1, B1, C1, D1, A0, B0, C0, D0

OUTPUTS: S1, S0, COUT

ATTRIBUTES:

INIT0: hexadecimal value (default: 16'h0000)

INIT1: hexadecimal value (default: 16'h0000)

INJECT1_0: "YES" (default), "NO"

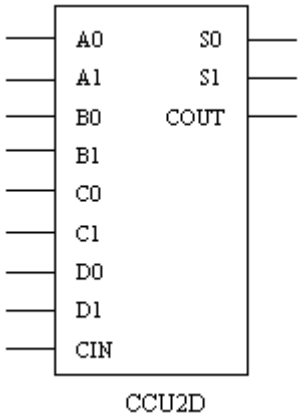
INJECT1_1: "YES" (default), "NO"

CCU2D

Carry Chain

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CIN, A1, B1, C1, D1, A0, B0, C0, D0

OUTPUTS: S1, S0, COUT

ATTRIBUTES:

INIT0: hexadecimal value (default: 16'h0000)

INIT1: hexadecimal value (default: 16'h0000)

INJECT1_0: "YES" (default), "NO"

INJECT1_1: "YES" (default), "NO"

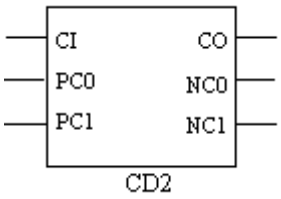
CD2

Combinational Logic for 2-Bit Down-Counter

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L

- Platform Manager
- Platform Manager 2



INPUTS: CI, PC0, PC1

OUTPUTS: CO, NC0, NC1

Description

This primitive realizes the combinational logic needed to implement a 2-bit down-counter using ripple elements.

Note

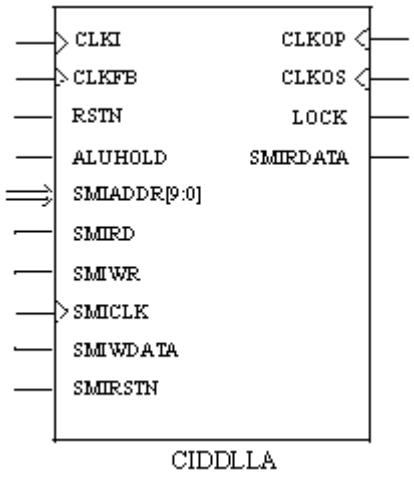
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

CIDDLLA

Clock Injection Delay Removal

Architectures Supported:

- LatticeECP2/M
- LatticeSC/M



INPUTS: CLKI, CLKFB, RSTN, ALUHOLD, SMIADDR9, SMIADDR8, SMIADDR7, SMIADDR6, SMIADDR5, SMIADDR4, SMIADDR3, SMIADDR2, SMIADDR1, SMIADDR0, SMIRD, SMIWR, SMICLK, SMIWDATA, SMIRSTN

OUTPUTS: CLKOP, CLKOS, LOCK, SMIRDATA

ATTRIBUTES:

CLKOP_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_PHASE: 360 + (0, 11, 22, 45) (default: 360)

CLKOS_FPHASE: 0 (default), 11, 22, 45

CLKOP_DUTY50: "DISABLED" (default), "ENABLED"

CLKOS_DUTY50: "DISABLED" (default), "ENABLED"

CLKI_DIV: 1 (default), 2, 4

CLKOS_DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

CLKOS_FDEL_ADJ: "DISABLED" (default), "ENABLED"

ALU_LOCK_CNT: integers 3~15 (default: 3)

ALU_UNLOCK_CNT: integers 3~15 (default: 3)

GLITCH_TOLERANCE: integers 0~7 (default: 2 for LatticeECP2/M; 0 for LatticeSC/M)

ALU_INIT_CNTVAL: 0 (default), 4, 8, 12, 16, 32, 48, 64, 72

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

(*LatticeSC/M only*) SMI_OFFSET: 0x400~0x7FF (default: 12'h410)

(*LatticeSC/M only*) MODULE_TYPE: "CIDDLLA"

(*LatticeSC/M only*) IP_TYPE: "CIDDLLA"

Description

CIDLLA removes the clock tree delay, aligning the external feedback clock to the reference clock. It has a single output coming from the fourth delay block. It features include clock tree insertion removal, N*Tcyc=4*Tdel + Tinj, lock achieved starting from minimum delay, and when it goes through all delay stages, the minimum frequency is 1/(4*Tdel). Its requirements include external feedback only, that you must use all delay cells, a maximum frequency of 700MHz, and a minimum frequency of 100MHz.

For more information, refer to the following technical notes on the Lattice web site:

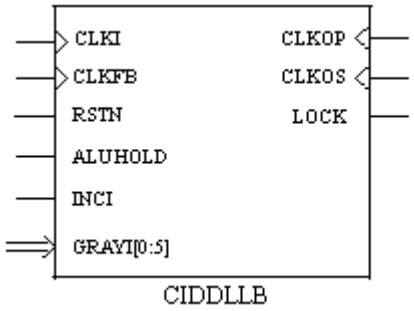
- TN1098 - LatticeSC sysCLOCK and PLL/DLL User's Guide
- TN1103 - LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide

CIDDLLB

Clock Injection Delay Removal

Architectures Supported:

- LatticeECP3



INPUTS: CLKI, CLKFB, RSTN, ALUHOLD, INCI, GRAYI5, GRAYI4, GRAYI3, GRAYI2, GRAYI1, GRAYI0

OUTPUTS: CLKOP, CLKOS, LOCK

ATTRIBUTES:

CLKOP_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_FPHASE: 0 (default), 11, 22, 33, 45, 56, 67, 78, 90, 101, 112, 123, 135, 146, 157, 169, 191, 202, 214, 225, 236, 247, 259, 281, 292, 304, 315, 326, 337, 349

CLKI_DIV: 1 (default), 2, 4

CLKOS_DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

ALU_LOCK_CNT: integers 3~15 (default: 3)

ALU_UNLOCK_CNT: integers 3~15 (default: 3)

GLITCH_TOLERANCE: integers 0~7 (default: 2)

ALU_INIT_CNTVAL: integers 0~31 (default: 0)

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

CLKOP_DUTY50: "DISABLED" (default), "ENABLED"

CLKOS_DUTY50: "DISABLED" (default), "ENABLED"

DEL0_GRAY: "DISABLED" (default), "ENABLED"

DEL1_GRAY: "DISABLED" (default), "ENABLED"

DEL2_GRAY: "DISABLED" (default), "ENABLED"

DEL3_GRAY: "DISABLED" (default), "ENABLED"

DEL4_GRAY: "DISABLED" (default), "ENABLED"

Description

CIDDLLB specifies the Clock Injection Delay Removal operation mode for the general purpose DLL (GDLL). In this mode, the feedback connection is supported and the CLKFB is captured on the CIDDLLB primitive.

Port Description

Table 602:

Port Name	Optional	Logical Capture Port Name
ALUHOLD	YES	HOLD
GRAYI[5:0]	YES	GRAY_IN[5:0]
INCI	YES	INC_IN
RSTN	YES	RSTN
CLKFB	NO	CLKFB
CLKI	NO	CLKI
CLK90	YES	CLK90
CLKOP	NO	CLKOP
CLKOS	YES	CLKOS
LOCK	NO	LOCK

For more information, refer to the following technical note on the Lattice web site:

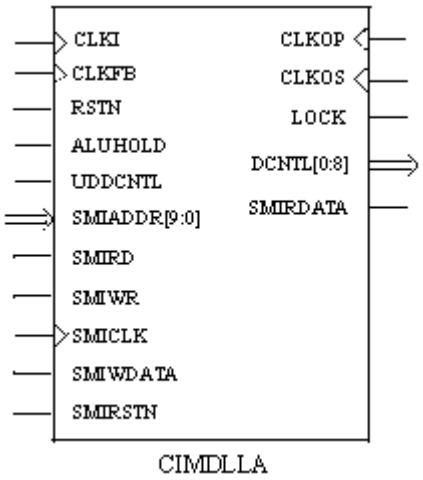
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide

CIMDLLA

Clock Injection Match

Architectures Supported:

- LatticeSC/M



INPUTS: CLKI, CLKFB, RSTN, ALUHOLD, UDDCNTL, SMIADDR9, SMIADDR8, SMIADDR7, SMIADDR6, SMIADDR5, SMIADDR4, SMIADDR3, SMIADDR2, SMIADDR1, SMIADDR0, SMIRD, SMIWR, SMICLK, SMIWDATA, SMIRSTN

OUTPUTS: CLKOP, CLKOS, LOCK, DCNTL0, DCNTL1, DCNTL2, DCNTL3, DCNTL4, DCNTL5, DCNTL6, DCNTL7, DCNTL8, SMIRDATA

ATTRIBUTES:

CLKOS_FPHASE: 0 (default), 11, 22, 45

CLKOS_DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

ALU_LOCK_CNT: integers 3~15 (default: 3)

ALU_UNLOCK_CNT: integers 3~15 (default: 3)

GLITCH_TOLERANCE: integers 0~7 (default: 0)

DCNTL_ADJVAL: integers -127~127 (default: 0)

SMI_OFFSET: 0x400~0x7FF (default: 12'h410)

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

CLKOS_FDEL_ADJ: "DISABLED" (default), "ENABLED"

MODULE_TYPE: "CIMDLLA"

IP_TYPE: "CIMDLLA"

Description

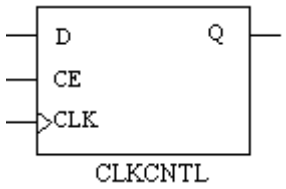
CIMDLLA matches the clock injection to one delay cell. This allows other inputs to take the registered ALU outputs and negate the clock injection delay. Its features include single clock output, lock achieved starting from minimum delay, output control bits, and allowance for +/- delay on these output control bits. Its requirements include external feedback (CLKOP) only, a maximum frequency of 700MHz, a minimum frequency of100MHz, and a maximum delay compensation of 3.9ns.

CLKCNTL

Clock Controller

Architectures Supported:

- LatticeSC/M



INPUTS: D, CE, CLK

OUTPUT: Q

ATTRIBUTES:

CLKMODE: "ECLK" (default), "SCLK"

Description

The CLKCNTL is the post-amble detect circuit required for the DQS input. The DQS generation will use the DELAY, TRDLLA and the CLKCNTL primitives.

The CLKCNTL primitive's instantiation rules allow the CLK input to be fed via secondary (local) routing paths, rather than constraining the routing to be via the Edge Clock tree. The Edge Clock tree is optimized for minimum skew rather than minimum delay. Although the Edge Clock delay is not a problem at the DDR/DDR2 clock frequencies originally targeted (300 MHz), the

CLKCNTL is capable of operating at much higher frequencies (beyond 1 GHz). If the CLK input path utilizes the faster local routing resources, it is capable of properly gating an 800 MHz clock, as is required for testing of DDR3 memory devices. This requires changes to the mapper.

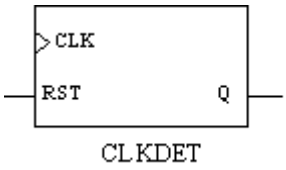
The CLKMODE attribute is supported for the CLKCNTL primitive. The legal values are ECLK (default) and SCLK.

CLKDET

Clock Detect

Architectures Supported:

- LatticeSC/M



INPUTS: CLK, RST

OUTPUT: Q

Truth Table

Table 603:

INPUTS		OUTPUTS
CK	RST	Q
X	1	0
↑	0	1

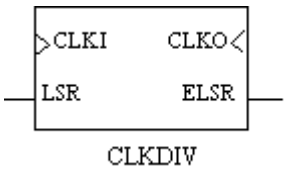
X = Don’t care

CLKDIV

Clock Divider

Architectures Supported:

- LatticeSC/M



INPUTS: CLKI, LSR

OUTPUTS: CLKO, ELSR

ATTRIBUTES:

DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

Description

Clock divider. Refer to the following technical note on the Lattice web site for more details.

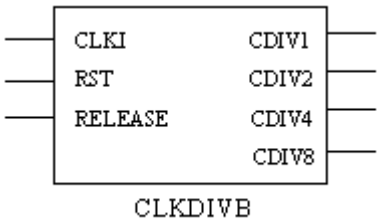
- TN1098 - LatticeSC sysCLOCK and PLL/DLL User's Guide

CLKDIVB

Clock Divider

Architectures Supported:

- LatticeECP2/M
- LatticeECP3
- LatticeXP2



INPUTS: CLKI, RST, RELEASE

OUTPUTS: CDIV1, CDIV2, CDIV4, CDIV8

ATTRIBUTES:

GSR: "DISABLED" (default), "ENABLED"

Description

Clock divider. See the following table for port description.

Table 604:

Port Name	I/O	Definition
RELEASE	I	Asserting the RELEASE signal releases the divided outputs, synchronous to selected input source.
RST	I	Asserting the RST signal forces CDIV1 low synchronously, and forces CDIV2, CDIV4 and CDIV8 low asynchronously. De-asserting RST synchronously allows all outputs to toggle.
CLKI	I	Input clock.
CDIV1	O	Divide by 1 output port.
CDIV2	O	Divide by 2 output port.
CDIV4	O	Divide by 4 output port.
CDIV8	O	Divide by 8 output port.

Refer to the following technical notes on the Lattice web site for more details.

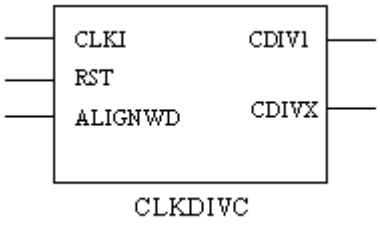
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide
- TN1126 - LatticeXP2 sysCLOCK PLL Design and Usage Guide
- TN1103 - LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide

CLKDIVC

Clock Divider

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLKI, RST, ALIGNWD

OUTPUTS: CDIV1, CDIVX

ATTRIBUTES:

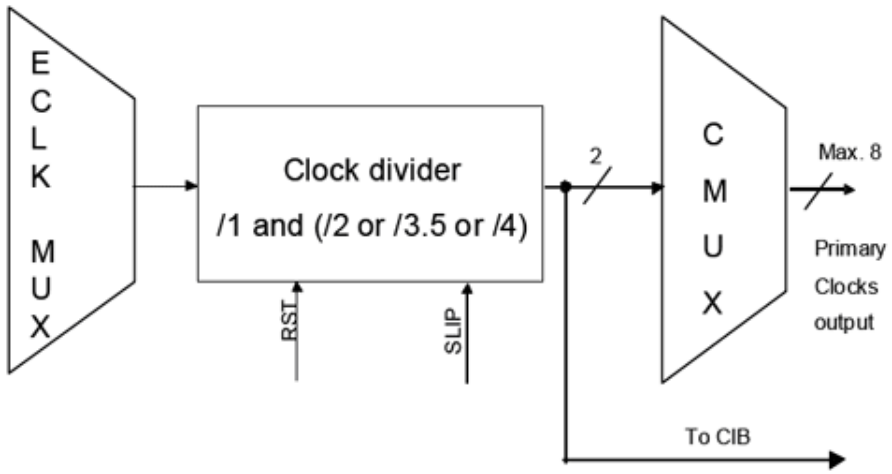
GSR: "DISABLED" (default), "ENABLED"

DIV: 2.0 (default), 3.5, 4.0

Description

A MachXO2/Platform Manager 2 device contains four CLKDIV with 1200 LUTs or more. The CLKDIV, or “clock divider” block, generates clock outputs one half, divided by three and a half or one quarter of the frequency of the input clock. It also generates an output clock of the same frequency as the input clock. All the outputs match input to output delay.

CLKDIV takes its inputs from the outputs of ECLK muxes. The divided outputs of the CLKDIV drive the primary clock center muxes directly and are also available on CIB ports for distribution to routing or secondary high fan out nets. The figure below represents the block diagram of CLKDIV:



The table below shows CLKDIVC IO description.

Table 605:

Port Name	I/O	Unused Nets	Description
RST	I	Tie low	Asserting the RST signal asynchronously forces all outputs low. De-asserting RST synchronously allows all outputs to toggle.
CLKI	I	Tie low	Input clock.
ALIGNWD	I	Tie low	This signal is used for word alignment.
CDIV1	O	Dangle	Divide by 1 output port.
CDIVX	O	Dangle	Divide by 2.0, 3.5, or 4.0 output port.

Refer to the following technical note on the Lattice web site for more details.

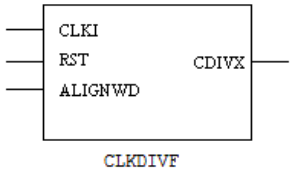
- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide

CLKDIVF

Clock Divider

Architectures Supported:

- ECP5



INPUTS: CLKI, RST, ALIGNWD

OUTPUTS: CDIVX

ATTRIBUTES:

GSR: "DISABLED" (default), "ENABLED"

DIV: 2.0 (default)

Description

Clock divider. See the following table for port description.

Table 606:

Port Name	I/O	Unused Nets	Description
RST	I	Tie low	Asynchronous, Active High Reset
CLKI	I	Tie low	Input clock.
ALIGNWD	I	Tie low	This signal is used for word alignment.
CDIVX	O	Dangle	Divide by output port.

CLKDIVG

Clock Divider

Architectures Supported:

- LIFMD
- LIFMDF



INPUTS: CLKI, RST, ALIGNWD

OUTPUTS: CDIVX

ATTRIBUTES:

GSR: "DISABLED" (default), "ENABLED"

DIV: 2.0 (default)

Description

Clock divider. See the following table for port description.

Table 607:

Port Name	I/O	Unused Nets	Description
RST	I	Tie low	Asynchronous, Active High Reset
CLKI	I	Tie low	Input clock.

Table 607:

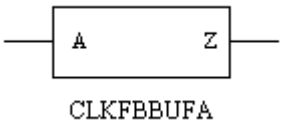
Port Name	I/O	Unused Nets	Description
ALIGNWD	I	Tie low	This signal is used for word alignment.
CDIVX	O	Dangle	Divide by output port.

CLKFBBUFA

Dummy Feedback Delay Between PLL clk Output and PLL fb Port

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUT: A

OUTPUT: Z

Description

The CLKFBBUFA is the dummy feedback path from the PLL clk output to the PLL feedback port to cancel out clock path variation over PVT.

The table below shows the IO description.

Table 608:

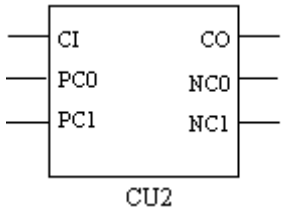
Port Name	I/O	Description
A	I	Clock input coming from the PLL CLKOP output
Z	O	Delayed output to the PLL fb port

CU2

Combinational Logic for 2-Bit Up-Counter

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: CI, PC0, PC1

OUTPUTS: CO, NC0, NC1

Description

This primitive realizes the combinational logic needed to implement a 2-bit up-counter using ripple elements.

When CI=0, NC[0:1]=PC[0:1] and CO=0

When CI=1, NC[0:1]=PC[0:1]+1, and CO=1 if PC[0:1]=11

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

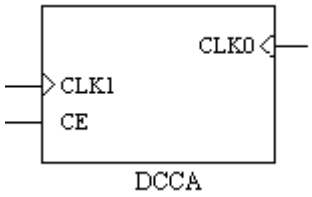
D

DCCA

Dynamic Quadrant Clock Enable/Disable

Architectures Supported:

- ECP5
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLKI, CE

OUTPUT: CLKO

Description

DCCA is the dynamic quadrant clock enable/disable primitive. In each quadrant, the dynamic quadrant clock enable/disable is available on the output of the center mux for the primary clocks CLK[5:0]. The dynamic quadrant clock enable/disable feature lets the internal logic control the quadrant primary clock network. When a clock network is disabled, all the logic fed by the clock network does not toggle, reducing the overall power consumption of the device. You need to instantiate a primitive (DCCA) in order to control the enable/disable function.

The DCCA IO description is shown below.

Table 609:

Port Name	I/O	Unused Nets	Description
CLKI	I	Tie low	Input clock
CE	I	Tie high	Clock enable
CLKO	O	Dangle	Output clock

DCCA Usage with VHDL

Library Instantiation

```
library lattice;
use lattice.components.all;
```

Component Declaration

```
component DCCA
   port (CLKI : in std_logic;
         CE   : in std_logic;
         CLKO : out std_logic);
end component;
```

DCCA Instantiation

```
I1: DCCA
   port map (CLKI => CLKI;
             CE   => CE;
             CLKO => CLKO);
end component;
```

DCCA Usage with Verilog HDL

Component Declaration

```
module DCCA (CLKI,
             CE,
             CLKO);
input CLKI;
input CE;
output CLKO;
endmodule
```

DCCA Instantiation

```
DCCA I1 (.CLKI (CLKI),
         .CE   (CE),
         .CLKO (CLKO));
```

For more information, refer to the following technical note on the Lattice web site:

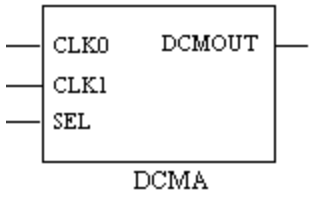
- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide

DCMA

Dynamic Clock Mux

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLK0, CLK1, SEL

OUTPUT: DCMOUT

Description

DCMA is a clock buffer incorporating a multiplexer function, whose output switches between the two clock inputs. It is not recommended that you route the PLL feedback signals through the multiplexer, because the PLL will loose lock.

DCMA IO description is shown in the below table.

Table 610:

Port Name	I/O	Unused Nets	Description
CLK0	I	Tie low	Input clock
CLKI	I	Tie low	Input clock
SEL	I	Tie low	Clock select from CIB
DCMOUT	O	Dangle	Output from the primary clock mux

DCMA Usage with VHDL

Library Instantiation

```
library lattice;
use lattice.components.all;
```

Component Declaration

```
component DCMA
   port (CLK0   : in std_logic;
         CLK1   : in std_logic;
         SEL    : in std_logic;
         DCMOUT :out std_logic);
end component;
```

DCMA Instantiation

```
I1: DCMA
   port map (CLK0 => CLK0,
             CLK1 => CLK1,
             SEL => SEL,
             DCMOUT => DCMOUT);
end component;
```

DCMA Usage with Verilog HDL

Component Declaration

```
module DCMA (CLK0, CLK1, SEL, DCMOUT);
input CLK0;
input CLK1;
input SEL;
output DCMOUT;
endmodule
```

DCMA Instantiation

```
DCMA I1 (.CLK0 (CLK0);
         .CLK1 (CLK1);
         .SEL (SEL);
         .DCMOUT (DCMOUT));
```

For additional information, see Lattice technical note on the web site:

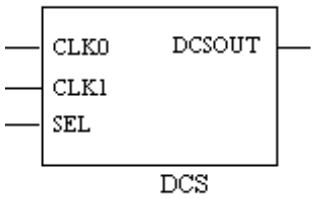
- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide

DCS

Dynamic Clock Selection

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2



INPUTS: CLK0 CLK1 SEL

OUTPUT: DCSOUT

ATTRIBUTES:

DCSMODE: "NEG" (default), "POS", "HIGH_LOW", "HIGH_HIGH", "LOW_LOW", "LOW_HIGH","CLK0", "CLK1"

Note

For LatticeECP/EC, LatticeECP2/M, LatticeXP, and LatticeXP2 devices, the DCSMODE default has been changed from POS to NEG since ispLEVER 7.0 SP2. If your pre-7.0SP2 design included the DCS macro and you didn't specify the DCSMODE value, the design may behave differently in Diamond. To avoid the issue, you can manually set DCSMODE to POS in your old design.

Description

DCS is a global clock buffer incorporating a smart multiplexer function that takes two independent input clock sources and avoids glitches or runt pulses on the output clock, regardless of where the enable signal is toggled. Located in pairs at the center of each edge, there are eight DCS primitives per device.

Table 611:

Function	Pins
Input Clock Select	SEL
Primary Clock Input 0	CLK0
Primary Clock Input 1	CLK1
To Primary Clock Mid-Mux	DCSOUT

For each DCS:

- Inputs are from primary clocks at PLC routing block (one branch per DCS).
- Selects are from a routing block's LUT port (A0, B0, and so on).
- Outputs are connected to the DCS output sources of the feedline muxes.

The outputs of the DCS then reach primary clock distribution via the feedlines. The connections from CIB to DCS are shown in the table below. The selected

CIB interfaces to a PIC and is located at the center of each bank and next to the mid-muxes. Note that the CIB to DCS connections are merged with the existing PIC-CIB connections at that CIB. It is up to the hardware designer to select the most convenient CIB.

CIB to DCS Connections

Table 612:

CIB	DCS
C7	SEL
CLK0(jclk0)	CLK0
CLK1(jclk2)	CLK1

DCSMODE Values

Table 613:

Attribute Name	Description	Output		Value	DCS Fuse Settings				
		SEL=0	SEL=1		0	1	2	3	4
DCS MODE	Rising edge triggered, latched state is high	CLK0	CLK1	POS	1	0	0	0	0
	Falling edge triggered, latched state is low	CLK0	CLK1	NEG	0	0	0	0	0
	Sel is active high, Disabled output is low	0	CLK1	HIGH_LOW	0	1	0	0	0
	Sel is active high, Disabled output is high	1	CLK1	HIGH_HIGH	1	1	0	0	0
	Sel is active low, Disabled output is low	CLK0	0	LOW_LOW	0	0	1	0	0
	Sel is active low, Disabled output is high	CLK0	1	LOW_HIGH	1	0	1	0	0
	Buffer for CLK0	CLK0	CLK0	CLK0	0	0	1	0	1
	Buffer for CLK1	CLK1	CLK1	CLK1	0	1	0	1	0

For additional information, see Lattice technical notes on the web site:

- TN1178 - LatticeECP3 sysCLOCK and PLL/DLL Design and Usage Guide
- TN1098 - LatticeSC sysCLOCK and PLL/DLL User's Guide
- TN1103 - LatticeECP2 sysCLOCK PLL/DLL Design and Usage Guide
- TN1126 - LatticeXP2 sysCLOCK PLL Design and Usage Guide

- TN1049 - LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide

Note

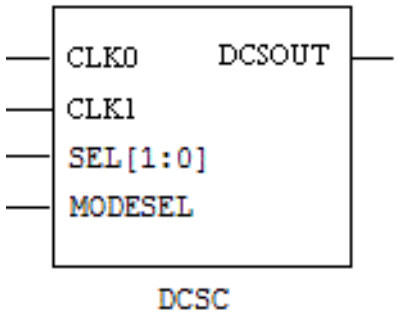
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

DCSC

Dynamic Clock Selection

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: CLK0 CLK1, SEL1, SEL0, MODESEL

OUTPUT: DCSOUT

ATTRIBUTES:

DCSMODE: "NEG" (default), "POS", "HIGH_LOW", "HIGH_HIGH", "LOW_LOW", "LOW_HIGH","CLK0", "CLK1"

Description

DCS is a 2-to-1 clock mux. Two modes exist. The glitchless mode switches between two clock sources (CLK0 and CLK1). The other mode added allows switching between the two clock sources asynchronously, based on the Sel signals. The second mode can produce a glitch on the output clock (DCSOUT), since the Sel signals could be changed at the same time the input clocks switching.

In the first mode (glitchless mode), there is a limitation on the circuit. If one of the clock sourcse is at a static H or L, the DCS will not make the switch. This

is because the DCS has to look at the two sources to synchronize the switch in order to not produce a glitch.

If the select inputs change from selecting one source (such as [01] to select CLK0), to selecting another source (such as [10] to select CLK1), where CLK1 is connected to GND, you can cycle the select input from [10] -> [00] (a few cycles) -> [01] to allow the switch from active clock (CLK0) to GND without glitching the output.

Alternatively, you can select non-glitchless mode (MODESEL = 1), and make the switch on the inputs. This could also make the output DCSOUT to switch from active clock (CLK0) to GND, but it could create a glitch when it makes the switch.

Table 614:

Function	Pins
Input Clock Select 0	SEL0
Input Clock Select 1	SEL1
Primary Clock Input 0	CLK0
Primary Clock Input 1	CLK1
Selects glitchless ("0") or non-glitchless ("1") behavior	MODESEL
To Primary Clock Mid-Mux	DCSOUT

DCSMODE Values When MODESEL = 0

Table 615:

Attribute Name	Operation	Default Value	Affected Fuses
DCSMODE	See table below. These options are only available when operating in glitchless mode (CIB pin MODESEL='0').	"POS"	Mode

DCSMODE Attribute Table When CIB MODESEL='0'

Table 616:

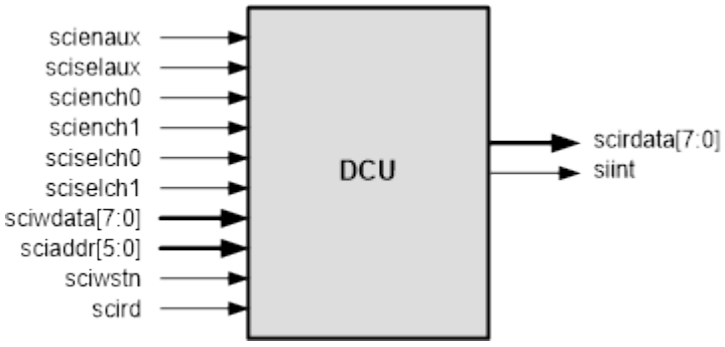
Attribute Name	Description	SEL[1:0]			Attribute Value	DCS Fuse Settings								
		2'b01	2'b10	2'b00/ 2'b11		0	1	2	3	4	5	6	7	8
DCSMODE Attributes	Falling Edge Triggered	Clk0	Clk1	0	NEG	0	0	0	0	0	0	0	0	0
	Rising Edge Triggered	Clk0	Clk1	1	POS	0	0	0	1	0	0	0	0	1
	Disabled Output is Low, Clk0	Clk0	0	0	CLK0_LOW	0	0	1	0	0	0	0	1	0
	Disabled Output is High, Clk0	Clk0	1	1	CLK0_HIGH	0	0	1	1	0	0	0	1	1
	Disabled Output is Low, Clk0	0	Clk1	0	CLK1_LOW	0	1	0	0	0	0	1	0	0
	Disabled Output is High, Clk0	1	Clk1	1	CLK1_HIGH	0	1	0	1	0	0	1	0	1
	Clk0 Buffered	Clk0	Clk0	0	CLK0	1	0	1	0	0	1	0	1	0
	Clk1 Buffered	Clk1	Clk1	1	CLK1	1	0	1	1	0	1	0	1	1
	Tie Low	0	0	0	LOW	1	1	1	0	0	1	1	1	0
	Tie High	1	1	1	HIGH	1	1	1	1	0	1	1	1	1
MODESEL = "1"	Non-Glitchless	Clk0	Clk1	0	-	x	x	x	x	x	x	x	x	x

DCUA

Dual channel unit SCI interface

Architectures Supported:

- ECP5



INPUTS: SCIENAUX, SCISELAUX, SCIENCH0, SCIENCH1, SCISELCH0, SCISELCH1, SCIWDATA[7:0], SCIADDR[5:0], SCIWSTN, SCIRD

OUTPUTS: SCIRDATA, SCIINT

Description

The SCI is an 8-bit asynchronous control interface to access the SERDES/ PCS channels and TX PLL inside the DCUA. The scien* and scisel* lines are used to select a resource. Then the sciwstn or scird ports are used to latch in the command to write or read 8-bit data from sciwdata or to scirdata respectively. In addition, there is a sciint port to allow an interrupt to be sent from any of the DCUA resources to the FPGA.

DCUA Port Definition

Table 617:

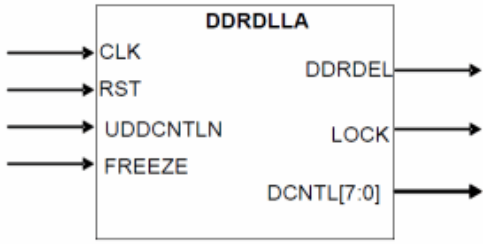
Port Name	I/O	Definition
SCIENAUX	I	(DCUA0) If d0txpll then signal, else unconnected.
SCISELAUX	I	(DCUA1) If d1txpll then signal, else unconnected.
SCIENCH0	I	(DCUA0) If d0ch0 then signal, else unconnected.
SCIENCH1	I	(DCUA0) If d0ch1 then signal, else unconnected.
SCISELCH0	I	(DCUA1) If d1ch0 then signal, else unconnected.
SCISELCH1	I	(DCUA1) If d1ch1 then signal, else unconnected.
SCIWDATA[7:0]	I	
SCIADDR[5:0]	I	
SCIWSTN	I	
SCIRD	I	
SCIRDATA	O	(For each DCUA) Always drive scirdata.
SCIINT	O	(For each DCUA) Always drive sciint.

DDRDLLA

90 degree delay for the DQS Input during a memory interface or the clock input for a generic DDR interface

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: CLK, RST, UDDCNTLN, FREEZE

OUTPUT: DDRDEL, LOCK, DCNTL7, DCNTL6, DCNTL5, DCNTL4, DCNTL3, DCNTL2, DCNTL1, DCNTL0

ATTRIBUTES:

FORCE_MAX_DELAY: "NO" (default), "YES"

LOCK_CYC: (For simulation only)

Description

The DDRDLL is used to generate a 90 degree delay for the DQS Input during a memory interface or the clock input for a generic DDR interface.

There are two DDRDLL modules per half the device. The DDRDLL outputs delay codes that are used in the DQSBUF elements to delay the DQS input or in the DLLDEL module to delay the input clock. DDRDLL by default will generate 90-degrees, but you may choose to generate 77, 88, 101 or 112 degrees as well.

DDRDLLA Port Definition

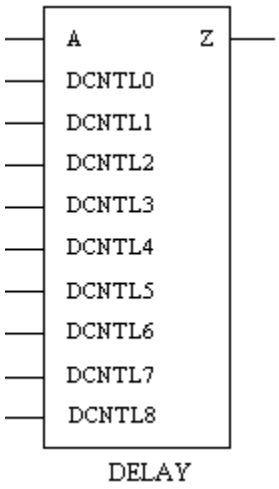
Table 618:

Port Name	I/O	Definition
CLK	I	Reference clock input to the DDRDLL. Should run at the same frequency as the clock to the delayed.
RST	I	Reset Input to the DDRDLL
UDDCNTLN	I	Update Control to update the delay code. When low the delay code out the DDRDLL is updated. Should not be active during a read or a write cycle.
FREEZE	I	When FREEZE goes high, it glitchlessly turns off the DLL internal clock and ring oscillator output clock. When FREEZE goes low, turns it back on.
DDRDEL	O	The delay codes from the DDRDLL to be used in DQSBUF or DLLDEL
LOCK	O	Lock output to indicate the DDRDLL has valid delay output
DCNTL[7:0]	O	The delay codes from the DDRDLL available for your IP.

DELAY

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: A, DCNTL0, DCNTL1, DCNTL2, DCNTL3, DCNTL4, DCNTL5, DCNTL6, DCNTL7, DCNTL8

OUTPUT: Z

Description

Sets the input delay for an input. You can choose either dynamic delay or the static delay. For more usage, see related technical notes or contact technical support.

Truth Table

Table 619:

INPUTS	OUTPUTS
A	Z
0	0
1	1

Note

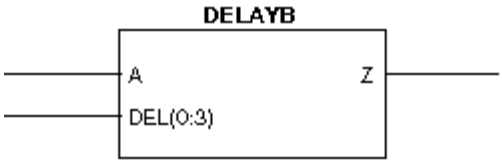
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

DELAYB

Dynamic Delay in PIO

Architectures Supported:

- LatticeECP2/M
- LatticeECP3
- LatticeXP2



INPUTS: A, DEL0, DEL1, DEL2, DEL3

OUTPUT: Z

Description

Data going to the DDR registers can be optionally delayed using the delay block. The DELAYB block receives a 4-bit delay value from the DLL. You can also choose to implement a fixed delay value instead of using the delay generated by the DLL. The various fixed delay choices can be made in the IPexpress software tool.

The DELAYB block can also be used to delay non-DDR inputs that use the input PIO register.

DELAYB Port Definition

Table 620:

Port Name	I/O	Definition
A	I	DDR input from sysIO buffer.
DEL [0:3]	I	Delay inputs.
Z	O	Output with delay.

Refer to the following technical notes for more details:

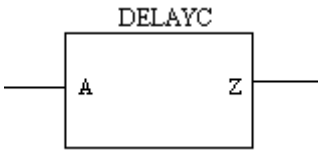
- TN1180 - LatticeECP3 High-Speed I/O Interface
- TN1105 - LatticeECP2/M High-Speed I/O Interface
- TN1138 - LatticeXP2 High-Speed I/O Interface

DELAYC

Fixed Delay in PIO

Architectures Supported:

- LatticeECP3



INPUT: A

OUTPUT: Z

DELAYC Port Definition

Table 621:

Port Name	I/O	Definition
A	I	DDR input from sysIO buffer.
Z	O	Output with delay.

Refer to the following technical note for more details:

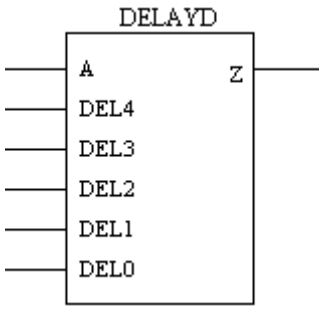
- TN1180 - LatticeECP3 High-Speed I/O Interface

DELAYD

Dynamic Delay for Bottom Bank

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUT: A, DEL4, DEL3, DEL2, DEL1, DEL0

OUTPUT: Z

Description

DELAYD is the dynamic delay for VPIC_RX, IOLA and IOLC cells in bottom side only. It can be used for x2, x4 and 7:1 applications. See the below table for its port description.

Table 622:

Port Name	I/O	Definition
A	I	Data input from IO buffer
DEL4, DEL3, DEL2, DEL1, DEL0	I	Dynamic delay inputs from CIB
Z	O	Output with delay

Refer to the following technical note for more details:

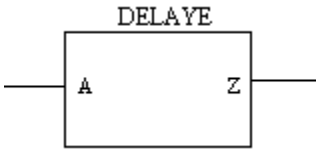
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

DELAYE

Fixed Delay in PIO

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUT: A

OUTPUT: Z

ATTRIBUTES:

DEL_MODE: "SCLK_ZEROHOLD", "ECLK_ALIGNED", "ECLK_CENTERED", "SCLK_ALIGNED", "SCLK_CENTERED", "USER_DEFINED" (default)

DEL_VALUE: "DELAY0" (default), "DELAY1", "DELAY2", ..., "DELAY31"

Description

DELAYE is the fix delay in PIO for all banks and all sides. It can be used for all IO registers and DDR types. See the below table for its IO port description.

Table 623:

Port Name	I/O	Definition
A	I	DDR input from sysIO buffer
Z	O	Output with delay

Refer to the following technical note for more details:

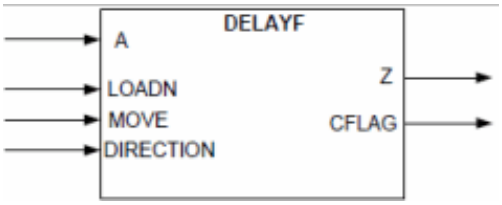
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

DELAYF

Delay

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUT: A, LOADIN, MOVE, DIRECTION

OUTPUT: Z, CFLAG

ATTRIBUTES:

DEL_MODE: "ECLK_ALIGNED", "ECLK_CENTERED", "ECLK_CENTERED_MIPI", "ECLK_CENTERED_SLVS", "SCLK_ALIGNED", "SCLK_CENTERED", "SCLK_ZEROHOLD", "USER_DEFINED" (default)

Note

"ECLK_CENTERED_MIPI" and "ECLK_CENTERED_SLVS" apply to LIFMD/F only.

DEL_VALUE: "DELAY0" (default), "DELAY1", "DELAY2", ..., "DELAY31"

Description

By default DELAYF is configured to factory delay settings based on the clocking structure. You can overwrite the DELAY setting using the MOVE and DIRECTION control inputs. The LOADN will reset the delay back to the default value.

Table 624:

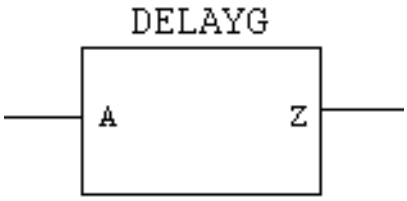
Port Name	I/O	Definition
A	I	Data input from pin or output register block
LOADIN	I	“0” on LOADN will reset to default Delay setting
MOVE	I	“Pulse” on MOVE will change delay setting. DIRECTION will be sampled at “falling edge” of MOVE.
DIRECTION	I	“1” to decrease delay & “0” to increase delay
Z	O	Delayed data to input register block or to pin
CFLAG	O	Flag indicating the delay counter has reached max (when moving up) or min (when moving down) value.

DELAYG

Delay

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUT: A

OUTPUT: Z

ATTRIBUTES:

DEL_MODE: "ECLK_ALIGNED", "ECLK_CENTERED", "ECLK_CENTERED_MIPI", "ECLK_CENTERED_SLVS", "SCLK_ALIGNED", "SCLK_CENTERED", "SCLK_ZEROHOLD", "USER_DEFINED" (default)

Note

"ECLK_CENTERED_MIPI" and "ECLK_CENTERED_SLVS" apply to LIFMD/F only.

DEL_VALUE: "DELAY0" (default), "DELAY1", "DELAY2", ..., "DELAY31"

Description

DELAYG is configured to factory delay settings based on the clocking structure. User cannot change the delay when using this module..

Table 625:

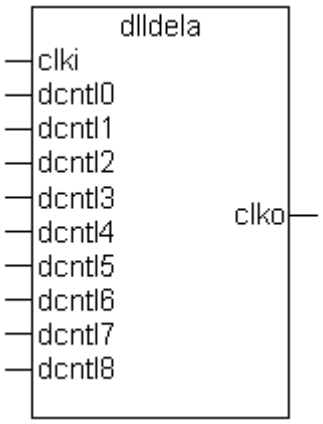
Port Name	I/O	Definition
A	I	Data input from pin or output register block.
Z	O	Delayed data to input register block or to pin.

DLLDELA

Slave Delay

Architectures Supported:

- LatticeECP2/M



INPUTS: CLKI, DCNTL0, DCNTL1, DCNTL2, DCNTL3, DCNTL4, DCNTL5, DCNTL6, DCNTL7, DCNTL8

OUTPUT: CLKO

Description

The Slave Delay Line is designed to generate desired delay in DDR/SPI4 applications. The delay control inputs (DCNTL[8:0]) are fed from the general purpose DLL outputs. The following table shows its port description.

Table 626:

Name	I/O	Description
CLKI	I	Clock input
DCNTL[8:0]	I	Delay control bits
CLKO	O	Clock output

For more details such as application examples, refer to the following technical note on the Lattice web site:

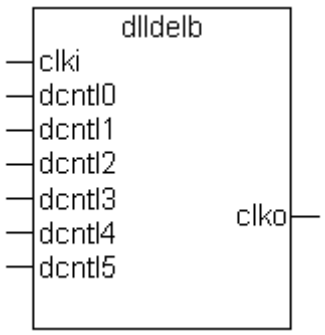
- TN1103 - LatticeECP2 sysCLOCK PLL/DLL Design and Usage Guide

DLLDELB

Slave Delay

Architectures Supported:

- LatticeECP3



INPUTS: CLKI, DCNTL5, DCNTL4, DCNTL3, DCNTL2, DCNTL1, DCNTL0

OUTPUT: CLKO

Description

DLLDELB is the LatticeECP3 slave delay line primitive. The DLLDELB port description is shown in the below table.

Table 627:

Port Name	I/O	Description
CLKI	I	Clock input
DCNTL0	I	Control bit 0 (hard wired to DLL)
DCNTL1	I	Control bit 1 (hard wired to DLL)
DCNTL2	I	Control bit 2 (hard wired to DLL)
DCNTL3	I	Control bit 3 (hard wired to DLL)
DCNTL4	I	Control bit 4 (hard wired to DLL)
DCNTL5	I	Control bit 5 (hard wired to DLL)
CLKO	O	Clock output

DLLDELB Usage with VHDL

```
COMPONENT DLLDELB
   PORT (CLKI    : IN  std_logic;
         DCNTL0  : IN  std_logic;
         DCNTL1  : IN  std_logic;
         DCNTL2  : IN  std_logic;
```

```
          DCNTL3 : IN  std_logic;
          DCNTL4 : IN  std_logic;
          DCNTL5 : IN  std_logic;
          CLKO   : OUT std_logic);
END COMPONENT;

begin
DLLDELAinst0: DLLDELB
   PORT MAP (
             CLKI   =>  clkisig,
             DCNTL0 =>  dcntl0sig,
             DCNTL1 =>  dcntl1sig,
             DCNTL2 =>  dcntl2sig,
             DCNTL3 =>  dcntl3sig,
             DCNTL4 =>  dcntl4sig,
             DCNTL5 =>  dcntl5sig,
             CLKO   =>  clkosig
            );
```

For more details such as application examples, refer to the following technical note on the Lattice web site:

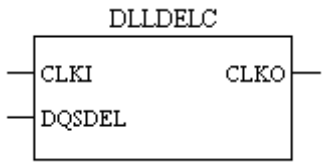
- TN1178 - LatticeECP3 sysCLOCK and PLL/DLL Design and Usage Guide

DLLDELC

Clock Shifting for ECLK or PCLK

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLKI, DQSDEL

OUTPUT: CLKO

ATTRIBUTES:

DEL_ADJ: "PLUS" (default), MINUS

DEL_VAL: integers 0~127 (PLUS), 1~128 (MINUS) (default: 0)

Description

DLLDELC is the generic input clock shifting using DQSDEL from DQSDLL. The DLLDELC port description is shown in the below table.

Table 628:

Port Name	I/O	Description
CLKI	I	clk input from I/O buffer
DQSDEL	I	Dynamic delay inputs from DQSDLLC
CLKO	O	Clock output with delay

For more details such as application examples, refer to the following technical note on the Lattice web site:

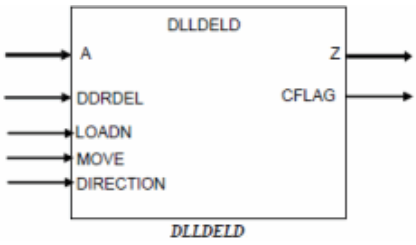
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

DLLDELD

Slave Delay

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: A, DDRDEL, LOADN, MOVE, DIRECTION

OUTPUT: Z, CFLAG

Description

DLLDELD is the delay element that receives code from DDRDLL for generic DDR. The DLLDELD port description is shown in the below table.

Table 629:

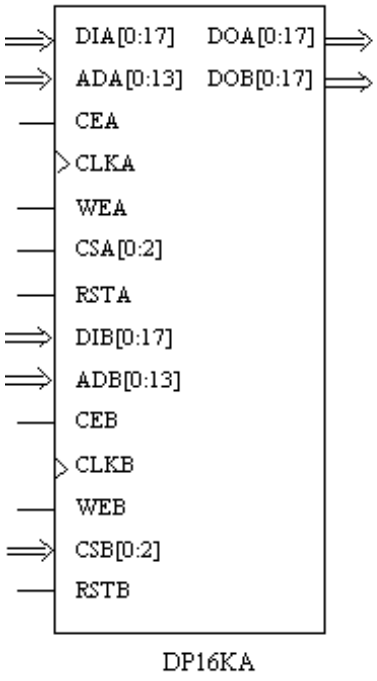
Port Name	I/O	Description
A	I	Clock Input
DDRDEL	I	Delay inputs from DDRDLL
LOADN	I	Used to reset back to 90 degrees delay.
MOVE	I	Pulse is required to change delay settings. The value on Direction will be sampled at the “falling edge” of MOVE.
DIRECTION	I	Indicates delay direction. “1” to decrease delay & “0” to increase delay.
CFLAG	O	Indicates the delay counter has reached max value when moving up or min value when moving down.
Z	O	Delayed Clock output

DP16KA

16K Dual Port Block RAM

Architectures Supported:

- LatticeSC/M



INPUTS: DIA0, DIA1, DIA2, DIA3, DIA4, DIA5, DIA6, DIA7, DIA8, DIA9, DIA10, DIA11, DIA12, DIA13, DIA14, DIA15, DIA16, DIA17, ADA0, ADA1, ADA2, ADA3, ADA4, ADA5, ADA6, ADA7, ADA8, ADA9, ADA10, ADA11, ADA12, ADA13, CEA, CLKA, WEA, CSA0, CSA1, CSA2, RSTA, DIB0, DIB1, DIB2, DIB3, DIB4, DIB5, DIB6, DIB7, DIB8, DIB9, DIB10, DIB11, DIB12, DIB13, DIB14, DIB15, DIB16, DIB17, ADB0, ADB1, ADB2, ADB3, ADB4, ADB5, ADB6, ADB7, ADB8, ADB9, ADB10, ADB11, ADB12, ADB13, CEB, CLKB, WEB, CSB0, CSB1, CSB2, RSTB

OUTPUTS: DOA0, DOA1, DOA2, DOA3, DOA4, DOA5, DOA6, DOA7, DOA8, DOA9, DOA10, DOA11, DOA12, DOA13, DOA14, DOA15, DOA16, DOA17, DOB0, DOB1, DOB2, DOB3, DOB4, DOB5, DOB6, DOB7, DOB8, DOB9, DOB10, DOB11, DOB12, DOB13, DOB14, DOB15, DOB16, DOB17

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9, 18 (default)

DATA_WIDTH_B: 1, 2, 4, 9, 18 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_A: any 3-bit binary value (default: 3’b000)

CSDECODE_B: any 3-bit binary value (default: 3’b000)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: (*Verilog*) 320'hXXX...X (80-bit hexadecimal value)
(*VHDL*) 0xXXX...X (80-bit hexadecimal value)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

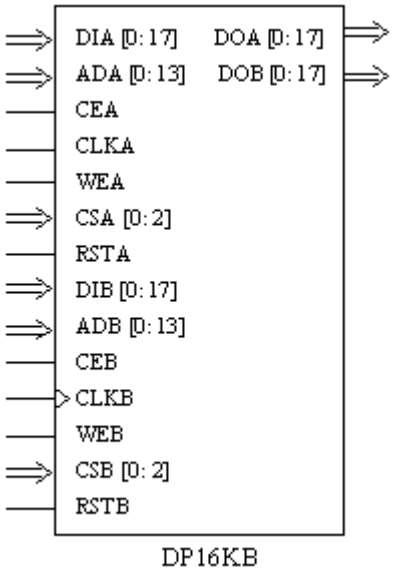
- TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices

DP16KB

True Dual Port Block RAM

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



DP16KB

INPUTS: DIA0, DIA1, DIA2, DIA3, DIA4, DIA5, DIA6, DIA7, DIA8, DIA9, DIA10, DIA11, DIA12, DIA13, DIA14, DIA15, DIA16, DIA17, ADA0, ADA1, ADA2, ADA3, ADA4, ADA5, ADA6, ADA7, ADA8, ADA9, ADA10, ADA11, ADA12, ADA13, CEA, CLKA, WEA, CSA0, CSA1, CSA2, RSTA, DIB0, DIB1,

DIB2, DIB3, DIB4, DIB5, DIB6, DIB7, DIB8, DIB9, DIB10, DIB11, DIB12, DIB13, DIB14, DIB15, DIB16, DIB17, ADB0, ADB1, ADB2, ADB3, ADB4, ADB5, ADB6, ADB7, ADB8, ADB9, ADB10, ADB11, ADB12, ADB13, CEB, CLKB, WEB, CSB0, CSB1, CSB2, RSTB

OUTPUTS: DOA0, DOA1, DOA2, DOA3, DOA4, DOA5, DOA6, DOA7, DOA8, DOA9, DOA10, DOA11, DOA12, DOA13, DOA14, DOA15, DOA16, DOA17, DOB0, DOB1, DOB2, DOB3, DOB4, DOB5, DOB6, DOB7, DOB8, DOB9, DOB10, DOB11, DOB12, DOB13, DOB14, DOB15, DOB16, DOB17

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9, 18 (default)

DATA_WIDTH_B: 1, 2, 4, 9, 18 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_A: any 3-bit binary value (default: 0b000)

CSDECODE_B: any 3-bit binary value (default: 0b000)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: 80-bit hexadecimal string (default: all zeros)

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

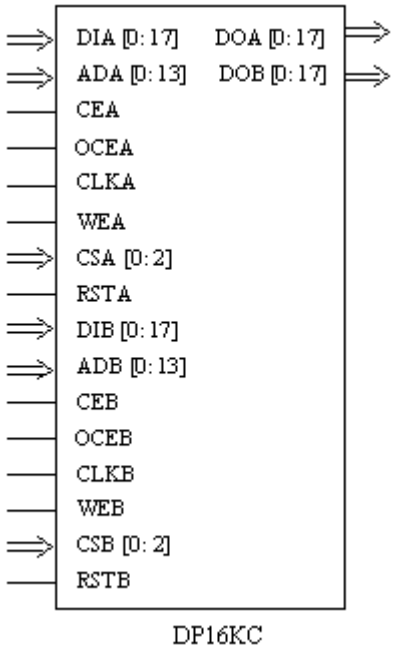
- TN1104 - LatticeECP2/M Memory Usage Guide

DP16KC

True Dual Port Block RAM

Architectures Supported:

- LatticeECP3



DP16KC

INPUTS: DIA17, DIA16, DIA15, DIA14, DIA13, DIA12, DIA11, DIA10, DIA9, DIA8, DIA7, DIA6, DIA5, DIA4, DIA3, DIA2, DIA1, DIA0, ADA13, ADA12, ADA11, ADA10, ADA9, ADA8, ADA7, ADA6, ADA5, ADA4, ADA3, ADA2, ADA1, ADA0, CEA, OCEA, CLKA, WEA, CSA2, CSA1, CSA0, RSTA, DIB17, DIB16, DIB15, DIB14, DIB13, DIB12, DIB11, DIB10, DIB9, DIB8, DIB7, DIB6, DIB5, DIB4, DIB3, DIB2, DIB1, DIB0, ADB13, ADB12, ADB11, ADB10, ADB9, ADB8, ADB7, ADB6, ADB5, ADB4, ADB3, ADB2, ADB1, ADB0, CEB, OCEB, CLKB, WEB, CSB2, CSB1, CSB0, RSTB

OUTPUTS: DOA17, DOA16, DOA15, DOA14, DOA13, DOA12, DOA11, DOA10, DOA9, DOA8, DOA7, DOA6, DOA5, DOA4, DOA3, DOA2, DOA1, DOA0, DOB17, DOB16, DOB15, DOB14, DOB13, DOB12, DOB11, DOB10, DOB9, DOB8, DOB7, DOB6, DOB5, DOB4, DOB3, DOB2, DOB1, DOB0

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9, 18 (default)

DATA_WIDTH_B: 1, 2, 4, 9, 18 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

CSDECODE_A: any 3-bit binary value (default: all zeros)

CSDECODE_B: any 3-bit binary value (default: all zeros)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: "0xXXX....X" (80-bit hex string) (default: all zeros)

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

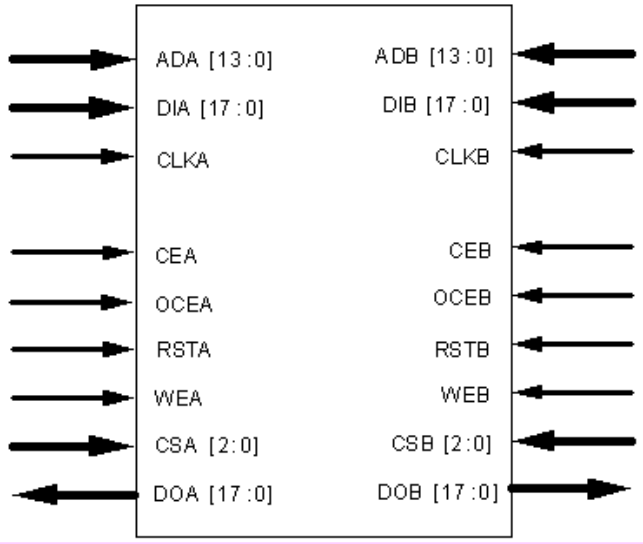
- TN1179 - LatticeECP3 Memory Usage Guide

DP16KD

True Dual Port Block RAM

Architectures Supported:

- ECP5



DP16KD

INPUTS: ADA13, ADA12, ADA11, ADA10, ADA9, ADA8, ADA7, ADA6, ADA5, ADA4, ADA3, ADA2, ADA1, ADA0, DIA17, DIA16, DIA15, DIA14, DIA13, DIA12, DIA11, DIA10, DIA9, DIA8, DIA7, DIA6, DIA5, DIA4, DIA3, DIA2, DIA1, DIA0, CLKA, CEA, OCEA, RSTA, WEA, CSA2, CSA1, CSA0, ADB13, ADB12, ADB11, ADB10, ADB9, ADB8, ADB7, ADB6, ADB5, ADB4, ADB3, ADB2, ADB1, ADB0, DIB17, DIB16, DIB15, DIB14, DIB13, DIB12, DIB11, DIB10, DIB9, DIB8, DIB7, DIB6, DIB5, DIB4, DIB3, DIB2, DIB1, DIB0, CLKB, CEB, OCEB, RSTB. WEB, CSB2, CSB1, CSB0

OUTPUTS: DOA17, DOA16, DOA15, DOA14, DOA13, DOA12, DOA11, DOA10, DOA9, DOA8, DOA7, DOA6, DOA5, DOA4, DOA3, DOA2, DOA1, DOA0, DOB17, DOB16, DOB15, DOB14, DOB13, DOB12, DOB11, DOB10, DOB9, DOB8, DOB7, DOB6, DOB5, DOB4, DOB3, DOB2, DOB1, DOB0

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9 (default)

DATA_WIDTH_B: 1, 2, 4, 9 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

CSDECODE_A: any 3-bit binary value (default: 3'b000)

CSDECODE_B: any 3-bit binary value (default: 3'b000)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH", "READBEFORE"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH", "READBEFORE"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

ASYNC_RESET_RELEASE: "SYNC" (default), "ASYNC"

INIT_DATA: "STATIC" (default), "DYNAMIC"

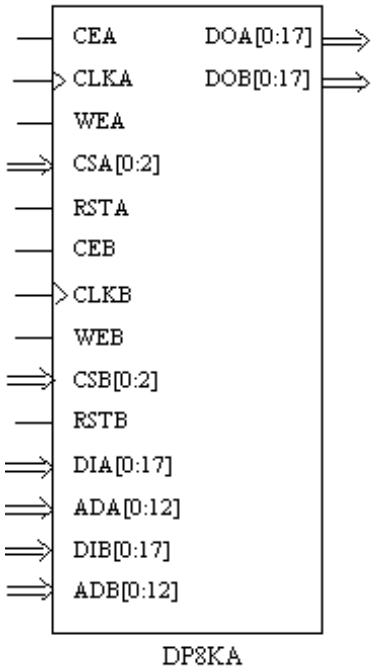
INITVAL_00 to INITVAL_1F: (*Verilog*) 320'hXXX...X (80-bit hex value)
(*VHDL*) 0xXXX...X (80-bit hex value)
Default: all zeros

DP8KA

8K Dual Port Block RAM

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: CEA, CLKA, WEA, CSA0, CSA1, CSA2, RSTA, CEB, CLKB, WEB, CSB0, CSB1, CSB2, RSTB, DIA0, DIA1, DIA2, DIA3, DIA4, DIA5, DIA6, DIA7, DIA8, DIA9, DIA10, DIA11, DIA12, DIA13, DIA14, DIA15, DIA16, DIA17, ADA0, ADA1, ADA2, ADA3, ADA4, ADA5, ADA6, ADA7, ADA8, ADA9, ADA10, ADA11, ADA12, DIB0, DIB1, DIB2, DIB3, DIB4, DIB5, DIB6, DIB7, DIB8, DIB9, DIB10, DIB11, DIB12, DIB13, DIB14, DIB15, DIB16, DIB17, ADB0, ADB1, ADB2, ADB3, ADB4, ADB5, ADB6, ADB7, ADB8, ADB9, ADB10, ADB11, ADB12

OUTPUTS: DOA0, DOA1, DOA2, DOA3, DOA4, DOA5, DOA6, DOA7, DOA8, DOA9, DOA10, DOA11, DOA12, DOA13, DOA14, DOA15, DOA16, DOA17, DOB0, DOB1, DOB2, DOB3, DOB4, DOB5, DOB6, DOB7, DOB8, DOB9, DOB10, DOB11, DOB12, DOB13, DOB14, DOB15, DOB16, DOB17

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9, 18 (default)

DATA_WIDTH_B: 1, 2, 4, 9, 18 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_A: any 3-bit binary value (default: 111)

CSDECODE_B: any 3-bit binary value (default: 111)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH"

INITVAL_00 to INITVAL_1F: (*Verilog*) 320'hXXX...X (80-bit hexadecimal value)
(*VHDL*) 0xXXX...X (80-bit hexadecimal value)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

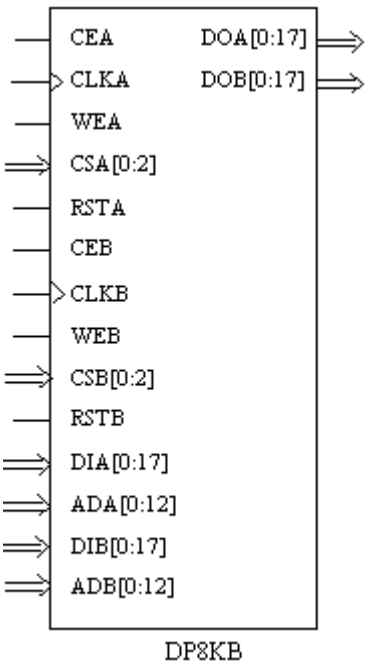
- TN1051 - Memory Usage Guide for LatticeECP/EC and LatticeXP Devices

DP8KB

8K Dual Port Block RAM

Architectures Supported:

- MachXO
- Platform Manager



INPUTS: CEA, CLKA, WEA, CSA0, CSA1, CSA2, RSTA, CEB, CLKB, WEB, CSB0, CSB1, CSB2, RSTB, DIA0, DIA1, DIA2, DIA3, DIA4, DIA5, DIA6, DIA7, DIA8, DIA9, DIA10, DIA11, DIA12, DIA13, DIA14, DIA15, DIA16,

DIA17, ADA0, ADA1, ADA2, ADA3, ADA4, ADA5, ADA6, ADA7, ADA8, ADA9, ADA10, ADA11, ADA12, DIB0, DIB1, DIB2, DIB3, DIB4, DIB5, DIB6, DIB7, DIB8, DIB9, DIB10, DIB11, DIB12, DIB13, DIB14, DIB15, DIB16, DIB17, ADB0, ADB1, ADB2, ADB3, ADB4, ADB5, ADB6, ADB7, ADB8, ADB9, ADB10, ADB11, ADB12

OUTPUTS: DOA0, DOA1, DOA2, DOA3, DOA4, DOA5, DOA6, DOA7, DOA8, DOA9, DOA10, DOA11, DOA12, DOA13, DOA14, DOA15, DOA16, DOA17, DOB0, DOB1, DOB2, DOB3, DOB4, DOB5, DOB6, DOB7, DOB8, DOB9, DOB10, DOB11, DOB12, DOB13, DOB14, DOB15, DOB16, DOB17

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9, 18 (default)

DATA_WIDTH_B: 1, 2, 4, 9, 18 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_A: any 3-bit binary value (default: all zeros)

CSDECODE_B: any 3-bit binary value (default: all zeros)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_1F: (*Verilog*) 320'hXXX...X (80-bit hex value)
(*VHDL*) 0xXXX...X (80-bit hex value)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

- TN1092 - MachXO Memory Usage Guide

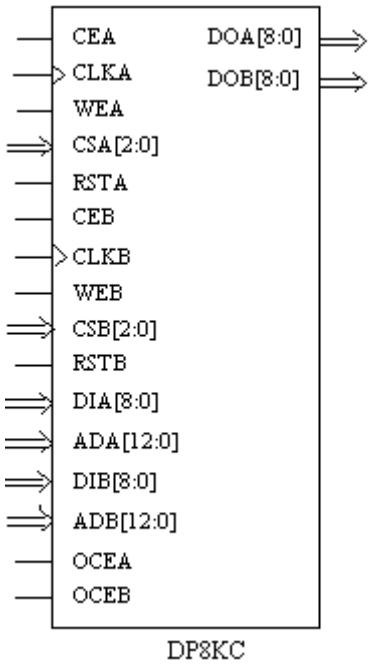
DP8KC

8K True Dual Port Block RAM

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D

- MachXO3L
- Platform Manager 2



INPUTS: CEA, CLKA, WEA, CSA2, CSA1, CSA0, RSTA, CEB, CLKB, WEB, CSB2, CSB1, CSB0, RSTB, DIA8, DIA7, DIA6, DIA5, DIA4, DIA3, DIA2, DIA1, DIA0, ADA12, ADA11, ADA10, ADA9, ADA8, ADA7, ADA6, ADA5, ADA4, ADA3, ADA2, ADA1, ADA0, DIB8, DIB7, DIB6, DIB5, DIB4, DIB3, DIB2, DIB1, DIB0, ADB12, ADB11, ADB10, ADB9, ADB8, ADB7, ADB6, ADB5, ADB4, ADB3, ADB2, ADB1, ADB0, OCEA, OCEB

OUTPUTS: DOA8, DOA7, DOA6, DOA5, DOA4, DOA3, DOA2, DOA1, DOA0, DOB8, DOB7, DOB6, DOB5, DOB4, DOB3, DOB2, DOB1, DOB0

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9 (default)

DATA_WIDTH_B: 1, 2, 4, 9 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

CSDECODE_A: any 3-bit binary value (default: 3'b000)

CSDECODE_B: any 3-bit binary value (default: 3'b000)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH", "READBEFORE"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH", "READBEFORE"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

ASYNC_RESET_RELEASE: "SYNC" (default), "ASYNC"

INIT_DATA: "STATIC" (default), "DYNAMIC"

INITVAL_00 to INITVAL_1F: (*Verilog*) 320'hXXX...X (80-bit hex value)
(*VHDL*) 0xXXX...X (80-bit hex value)
Default: all zeros

Description

The following table describes the I/O ports of the DP8KC primitive.

Table 630:

Port Name	I/O	Definition
CEA, CEB	I	Clock enable for port CLKA and CLKB
OCEA, OCEB	I	Output clock enable for port A and B
CLKA, CLKB	I	Clock for port A and B
RSTA, RSTB	I	Reset for port A and B
WEA, WEB	I	Write enable for port A and B
CSA[2:0], CSB[2:0]	I	Chip select for port A and B
DIA[8:0], DIB[8:0]	I	Input data port A and B (up to 9)
ADA[12:0], ADB[12:0]	I	Address bus port A and B (up to 13)
DOA[8:0], DOB[8:0]	O	Output data port A and B (up to 9)

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

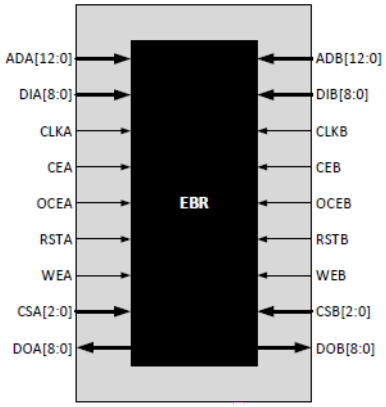
- TN1201 - Memory Usage Guide for MachXO2 Devices

DP8KE

True Dual Port EBR RAM

Architectures Supported:

- LIFMD
- LIFMDF



INPUTS: DIA8, DIA7, DIA6, DIA5, DIA4, DIA3, DIA2, DIA1, DIA0, ADA12, ADA11, ADA10, ADA9, ADA8, ADA7, ADA6, ADA5, ADA4, ADA3, ADA2, ADA1, ADA0, CEA, OCEA, CLKA, WEA, CSA2, CSA1, CSA0, RSTA, DIB8, DIB7, DIB6, DIB5, DIB4, DIB3, DIB2, DIB1, DIB0, ADB12, ADB11, ADB10, ADB9, ADB8, ADB7, ADB6, ADB5, ADB4, ADB3, ADB2, ADB1, ADB0, CEB, OCEB, CLKB, WEB, CSB2, CSB1, CSB0, RSTB OUTPUTS: DOA8, DOA7, DOA6, DOA5, DOA4, DOA3, DOA2, DOA1, DOA0, DOB8, DOB7, DOB6, DOB5, DOB4, DOB3, DOB2, DOB1, DOB0

OUTPUTS: DOA8, DOA7, DOA6, DOA5, DOA4, DOA3, DOA2, DOA1, DOA0, DOB8, DOB7, DOB6, DOB5, DOB4, DOB3, DOB2, DOB1, DOB0

ATTRIBUTES:

DATA_WIDTH_A: 1, 2, 4, 9 (default)

DATA_WIDTH_B: 1, 2, 4, 9 (default)

REGMODE_A: "NOREG" (default), "OUTREG"

REGMODE_B: "NOREG" (default), "OUTREG"

CSDECODE_A: any 3-bit binary value (default: 3’b000)

CSDECODE_B: any 3-bit binary value (default: 3’b000)

WRITEMODE_A: "NORMAL" (default), "WRITETHROUGH", "READBEFORE"

WRITEMODE_B: "NORMAL" (default), "WRITETHROUGH", "READBEFORE"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

ASYNC_RESET_RELEASE: "SYNC" (default), "ASYNC"

INIT_DATA: "STATIC" (default), "DYNAMIC"

INITVAL_00 to INITVAL_1F: (*Verilog*) 320'hXXX...X (80-bit hex value)
(*VHDL*) 0xXXX...X (80-bit hex value)
Default: all zeros

Description

The following table describes the I/O ports of the DP8KE primitive.

Table 631:

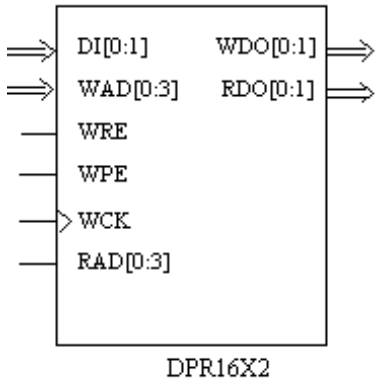
Port Name	Description
CEA, CEB	Clock Enables for Port CLKA and CLKB
OCEA, OCEB	Output Clock Enables for Port A and B
CLKA, CLKB	Clock for port A and port B
RSTA, RSTB	Reset for port A and port B
WEA, WEB	Write enable for port A and port B
CSA[2:0], CSB[2:0]	Chip Selects for port A and port B
DIA[8:0], DIB[8:0]	Input Data port A and port B
ADA[12:0], ADB[12:0]	Address Bus port A and port B
DOA [8:0], DOB[8:0]	Output Data Port A and port B
ADW[8:0]	Write Address
ADR[12:0]	Read Address

DPR16X2

Distributed Dual Port RAM

Architectures Supported:

- LatticeSC/M



INPUTS: DI0, DI1, WAD0, WAD1, WAD2, WAD3, WRE, WPE, WCK, RAD0, RAD1, RAD2, RAD3

OUTPUTS: WDO0, WDO1, RDO0, RDO1

ATTRIBUTES:

INITVAL: (*Verilog*) 64'hXXXXXXXX (16-bit hexadecimal value)
(*VHDL*) 0xXXXXXXXX (16-bit hexadecimal value)
Default: all zeros

GSR: "ENABLED" (default), "DISABLED"

Description

The DPR16X2 symbol represents a 16-word by 2-bit distributed dual port RAM. You can refer to the following technical note on the Lattice web site for port definition, attribute definition and usage.

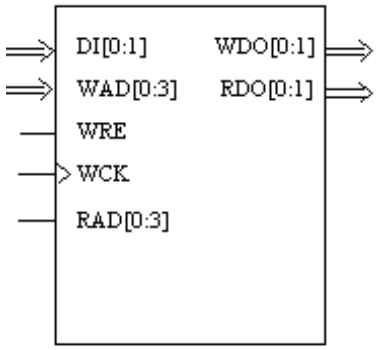
- TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices

DPR16X2B

Distributed Dual Port RAM

Architectures Supported:

- LatticeECP/EC
- LatticeXP
- MachXO
- Platform Manager



DPR16X2B

INPUTS: DI0, DI1, WAD0, WAD1, WAD2, WAD3, WRE, WCK, RAD0, RAD1, RAD2, RAD3

OUTPUTS: WDO0, WDO1, RDO0, RDO1

Description

You can refer to the following technical notes on the Lattice web site for port definition, attribute definition and usage.

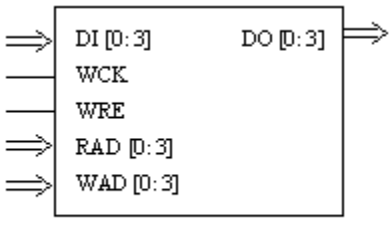
- TN1051 - Memory Usage Guide for LatticeECP/EC and LatticeXP Devices
- TN1092 - MachXO Memory Usage Guide

DPR16X4A

Distributed Pseudo Dual Port RAM

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



DPR16X4A

INPUTS: DI0, DI1, DI2, DI3, WCK, WRE, RAD0, RAD1, RAD2, RAD3, WAD0, WAD1, WAD2, WAD3

OUTPUTS: DO0, DO1, DO2, DO3

Description

PFU-based distributed Pseudo Dual-port RAM primitive. See Memory Primitives Overview for individual port description.

You can also refer to the following technical notes on the Lattice web site for port definition, attributes and usage.

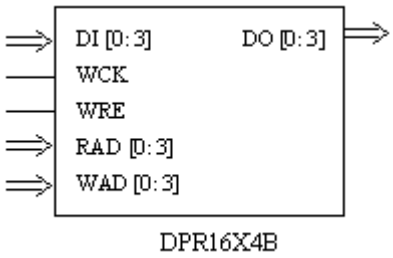
- TN1104 - LatticeECP2/M Memory Usage Guide
- TN1137 - LatticeXP2 Memory Usage Guide

DPR16X4B

Distributed Pseudo Dual Port RAM

Architectures Supported:

- LatticeXP2



INPUTS: WAD0, WAD1, WAD2, WAD3, DI0, DI1, DI2, DI3, WCK, WRE, RAD0, RAD1, RAD2, RAD3

OUTPUTS: DO0, DO1, DO2, DO3

ATTRIBUTES:

INITVAL: (*Verilog*) "64'hXXXXXXXXXXXXXXXX" (16-bit hex string)
(*VHDL*) "0xXXXXXXXXXXXXXXXX" (16-bit hex string)
Default: all zeros

Description

PFU-based distributed Pseudo Dual-port RAM primitive. See Memory Primitives Overview for individual port description.

You can also refer to the following technical note on the Lattice web site for port definition, attributes and usage.

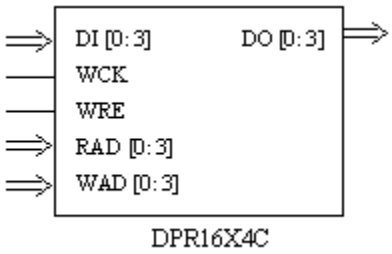
- TN1137 - LatticeXP2 Memory Usage Guide

DPR16X4C

Distributed Pseudo Dual Port RAM

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: DI3, DI2, DI1, DI0, WAD3,WAD2,WAD1,WAD0, WCK, WRE, RAD3,RAD2,RAD1,RAD0

OUTPUTS: DO3, DO2, DO1, DO0

ATTRIBUTES:

INITVAL: "0xXXXXXXXXXXXXXXXX" (16-bit hex string) (default: all zeros)

Description

PFU-based distributed Pseudo Dual-port RAM primitive. See Memory Primitives Overview for individual port description.

You can also refer to the following technical notes on the Lattice web site for port definition, attributes and usage.

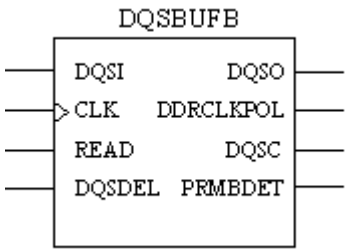
- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1179 - LatticeECP3 Memory Usage Guide

DQSBUFB

DDR DQS Buffer Used as DDR memory DQS generator

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: DQSI, CLK, READ, DQSDEL

OUTPUTS: DQSO, DDRCLKPOL, DQSC, PRMBDET

Description

This cell is used to indicate how many DDR I/Os need to be tied together, aligning the placement of the DDR cell. The input goes to the clock and the output goes to the clock on the DDR cell. For more usage, see related technical notes or contact technical support.

Refer to the following technical note on the Lattice web site for more details:

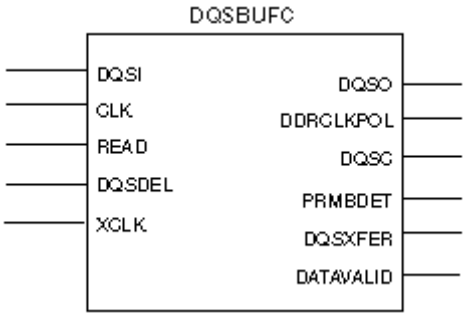
- TN1050 - LatticeECP/EC and LatticeXP DDR Usage Guide

DQSBUFC

DQS Delay Function and Clock Polarity Selection Logic

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: DQSI, CLK, XCLK, READ, DQSDEL

OUTPUTS: DQSO, DDRCLKPOL, DQSC, PRMBDET, DQSXFER, DATAVALID

ATTRIBUTES:

DQS_LI_DEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 4)

DQS_LI_DEL_ADJ: "MINUS" (default), "PLUS"

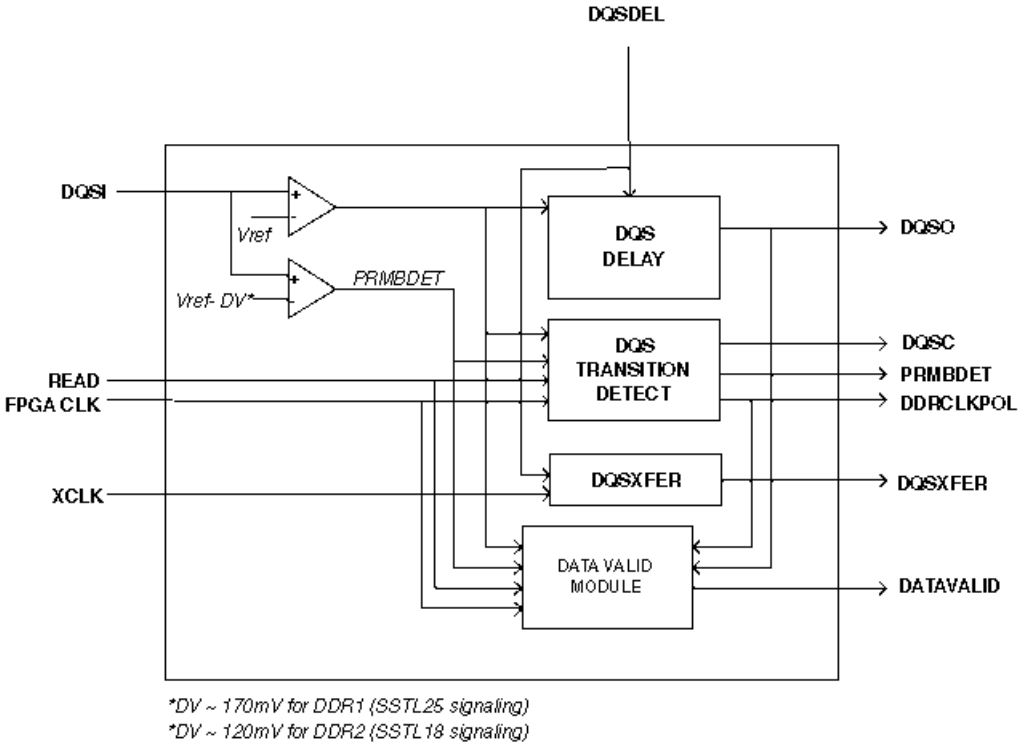
DQS_LO_DEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

DQS_LO_DEL_ADJ: "PLUS" (default), "MINUS"

Description

DQSBUFC implements the DQS delay and the DQS transition detector logic. The primitive is composed of the DQS Delay, the DQS Transition Detect and the DQSXFER block as shown in the following figure. This block inputs the DQS and delays it by 90 degrees. It also generates the DDR Clock Polarity and the DQSXFER signal. The preamble detect (PRMBDET) signal is generated from the DQSI input using a voltage divider circuit.

DQSBUFC Function



DQS Delay Block: The DQS Delay block receives the digital control delay line (DQSDEL) coming from one of the two DQSDLL blocks. These control signals are used to delay the DQSI by 90 degrees. DQSO is the delayed DQS and is connected to the clock input of the first set of DDR registers.

DQS Transition Detect: The DQS Transition Detect block generates the DDR Clock Polarity signal based on the phase of the FPGA clock at the first DQS transition. The DDR READ control signal and FPGA CLK inputs to this coming and should be coming from the FPGA core.

DQSXFER: This block generates the 90-degree phase shifted clock to for the DDR Write interface. The input to this block is the XCLK. You can choose to connect this either to the edge clock or the FPGA clocks. The DQSXFER is routed using the DQSXFER tree to all the I/Os spanned by that DQS.

Data Valid Module: The data valid module generates a DATAVALID signal. This signal indicates to the FPGA that valid data is transmitted out of the input DDR registers to the FPGA core.

The following table describes DQSBUFC I/O ports.

Table 632:

Port Name	I/O	Definition
DQSI	I	DQS strobe signal from memory
CLK	I	System CLK
READ	I	Read generated from the FPGA core
DQSDEL	I	DQS delay from the DQSDLL primitive
XCLK	I	Edge clock or system CLK
DQSO	O	Delayed DQS strobe signal, to the input capture register block
DQSC	O	DQS strobe signal before delay, going to the FPGA core logic
DDRCLKPOL	O	DDR clock polarity signal
PRMBDET	O	Preamble detect signal, going to the FPGA core logic
DQSXFER	O	90 degree shifted clock going to the output DDR register block
DTATVALID	O	Signal indicating transmission of valid data to the FPGA core

READ Pulse Generation

The READ signal to the DQSBUFC block is internally generated in the FPGA core. The READ signal goes high when the READ command to control the DDR-SDRAM is initially asserted. This precedes the DQS preamble by one cycle, yet may overlap the trailing bits of a prior read cycle. The DQS Detect circuitry of the LatticeECP2 device requires the falling edge of the READ signal to be placed within the preamble stage.

The preamble state of the DQS can be detected using the CAS latency and the round trip delay for the signals between the FPGA and the memory device. Note that the internal FPGA core generates the READ pulse. The rise of the READ pulse should coincide with the initial READ command of the Read Burst and need to go low before the Preamble goes high.

Refer to the following technical notes on the Lattice web site for more details:

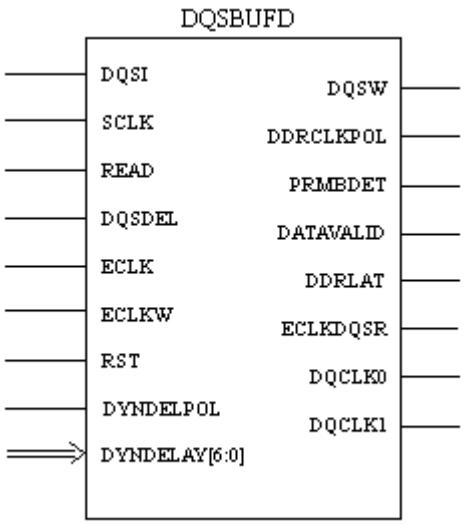
- TN1105 - LatticeECP2/M High-Speed I/O Interface
- TN1138 - LatticeXP2 High-Speed I/O Interface

DQSBUFD

DDR DQS Buffer Used for DDR3_MEM and DDR3_MEMGEN

Architectures Supported:

- LatticeECP3



INPUTS: DQSI, SCLK, READ, DQSDEL, ECLK, ECLKW, RST, DYNDELPOL, DYNDELAY6, DYNDELAY5, DYNDELAY4, DYNDELAY3, DYNDELAY2, DYNDELAY1, DYNDELAY0

OUTPUTS: DQSW, DDRCLKPOL, PRMBDET, DATAVALID, DDRLAT, ECLKDQSR, DQCLK0, DQCLK1

ATTRIBUTES:

DYNDEL_TYPE: "NORMAL" (default), "SHIFTED"

DYNDEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

DYNDEL_CNTL: "DYNAMIC" (default), "STATIC"

NRZMODE: (*EA only*) "DISABLED" (default), "ENABLED"

Description

DQSBUFD is the DDR DQS buffer used for DDR3_MEM (DDR3 memory mode) and DDR3_MEMGEN.

- E and EA: DDR3_MEM and DDR3_MEMGEN (left/right)
- EA: DDR3_MEMGEN (top) – only input side

The table below describes the I/O ports.

Table 633:

Signal	I/O	Description
DQSI	I	DQS input coming from the pad.
SCLK	I	System clock.

Table 633:

Signal	I/O	Description
READ	I	READ signal generated from the FPGA core.
RST	I	Reset input.
DQSDEL	I	Delay input from DQSDLL.
ECLK	I	Edge clock.
ECLKW	I	Edge clock used for the DDR write side.
DYNDELAY[6:0]	I	From user logic to DLL ADW & write clock generation.
DYNDELPOL	I	From user logic to DLL ADW & write clock generation. Will change the polarity of the clock depending on the clock frequency.
DQSW	O	DQS write clock.
DDRCLKPOL	O	DDR clock polarity signal.
PRMBDET	O	The preamble detect signal generated from the DQS signal going to the CIB. DQSI biased to go high when DQSI is tri-state.
DATAVALID	O	Signal indicating the transmission of valid data to the FPGA core.
DDRLAT	O	DDR latch control to input logic. Used to guarantee IDDRX2 gearing by selectively enabling a D flip-flop in the data path.
ECLKDQSR	O	Delay DQS used to capture the data.
DQCLK0	O	One clock edge, at half the frequency of ECLK, used in output gearing, 90 degree out of phase from DQCLK1.
DQCLK1	O	One clock edge, at the frequency of SCLK, used in output gearing.

Refer to the following technical note on the Lattice web site for more details:

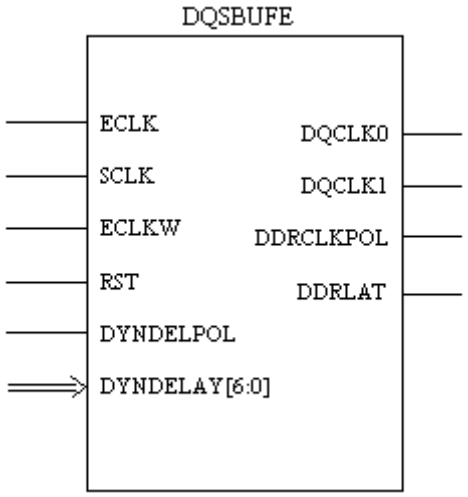
- TN1180 - LatticeECP3 High-Speed I/O Interface

DQSBUFE

DDR DQS Buffer Used for DDR_GENX2

Architectures Supported:

- LatticeECP3



INPUTS: ECLK, SCLK, ECLKW, RST, DYNDELPOL, DYNDELAY6, DYNDELAY5, DYNDELAY4, DYNDELAY3, DYNDELAY2, DYNDELAY1, DYNDELAY0

OUTPUTS: DQCLK0, DQCLK1, DDRCLKPOL, DDRLAT

ATTRIBUTES:

DYNDEL_TYPE: "NORMAL" (default), "SHIFTED"

DYNDEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

DYNDEL_CNTL: "DYNAMIC" (default), "STATIC"

Description

DQSBUFE is the DDR DQS buffer used for DDR_GENX2 (DDR generic mode in X2 gearing).

- E: DDR_GENX2 (left/right/top)

The table below describes the I/O ports.

Table 634:

Signal	I/O	Description
SCLK	I	System clock.
ECLK	I	Edge clock.
ECLKW	I	Edge clock used for the DDR write side.
RST	I	Reset input.
DYNDELAY[6:0]	I	From user logic to DLL ADW & write clock generation.

Table 634:

Signal	I/O	Description
DYNDELPOL	I	From user logic to DLL ADW & write clock generation. Will change the polarity of the clock depending on the clock frequency.
DDRCLKPOL	O	DDR clock polarity signal.
DDRLAT	O	DDR latch control to input logic. Used to guarantee IDDRX2 gearing by selectively enabling a D flip-flop in the data path.
DQCLK0	O	One clock edge, at half the frequency of ECLK, used in output gearing, 90 degree out of phase from DQCLK1.
DQCLK1	O	One clock edge, at half the frequency of SCLK, used in output gearing.

Refer to the following technical note on the Lattice web site for more details:

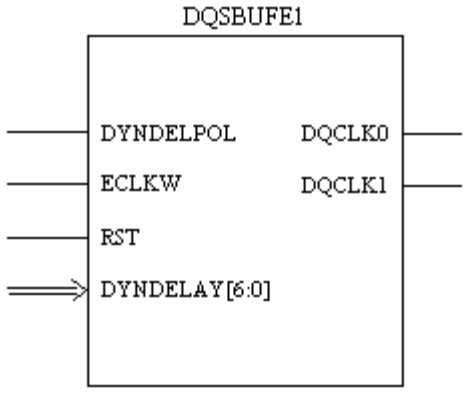
- TN1180 - LatticeECP3 High-Speed I/O Interface

DQSBUFE1

DDR DQS Buffer Used for DDR_GENX2

Architectures Supported:

- LatticeECP3



INPUTS: ECLKW, RST, DYNDELPOL, DYNDELAY6, DYNDELAY5, DYNDELAY4, DYNDELAY3, DYNDELAY2, DYNDELAY1, DYNDELAY0

OUTPUTS: DQCLK0, DQCLK1

ATTRIBUTES:

DYNDEL_TYPE: "NORMAL" (default), "SHIFTED"

DYNDEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

DYNDEL_CNTL: "DYNAMIC" (default), "STATIC"

Description

DQSBUFE1 is the DDR DQS buffer used for DDR_GENX2 (DDR generic mode in X2 gearing).

- EA: DDR_GENX2 (left/right)
- EA: DDR_GENX2 (top)

The table below describes the I/O ports.

Table 635:

Signal	I/O	Description
ECLKW	I	Edge clock used for the DDR write side.
RST	I	Reset input.
DYNDELAY[6:0]	I	From user logic to DLL ADW & write clock generation.
DYNDELPOL	I	From user logic to DLL ADW & write clock generation. Will change the polarity of the clock depending on the clock frequency.
DQCLK0	O	One clock edge, at half the frequency of ECLK, used in output gearing, 90 degree out of phase from DQCLK1.
DQCLK1	O	One clock edge, at half the frequency of SCLK, used in output gearing.

Refer to the following technical note on the Lattice web site for more details:

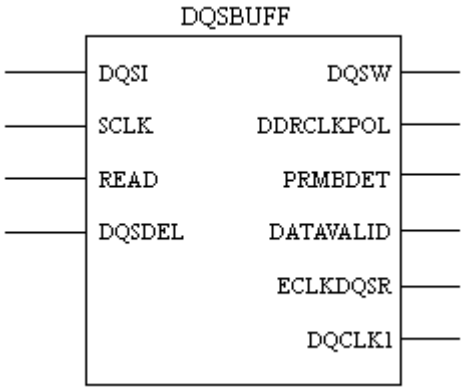
- TN1180 - LatticeECP3 High-Speed I/O Interface

DQSBUFF

DDR DQS Buffer Used for DDR_MEM, DDR2_MEM, and DDR2_MEMGEN

Architectures Supported:

- LatticeECP3



INPUTS: DQSI, SCLK, READ, DQSDEL

OUTPUTS: DQSW, DDRCLKPOL, PRMBDET, DATAVALID, ECLKDQSR, DQCLK1

Description

DQSBUFF is the DDR DQS buffer used for DDR_MEM (DDR memory mode), DDR2_MEM (DDR2 memory mode), and DDR2_MEMGEN.

- E and EA: DDR_MEM, DDR2_MEM, and DDR2_MEMGEN (left/right/top)

The table below describes the I/O ports.

Table 636:

Signal	I/O	Description
DQSI	I	DQS input coming from the pad.
SCLK	I	System clock.
READ	I	READ signal generated from the FPGA core.
DQSDEL	I	Delay input from DQSDLL.
DQSW	O	DQS write clock.
DDRCLKPOL	O	DDR clock polarity signal.
PRMBDET	O	The preamble detect signal generated from the DQS signal going to the CIB. DQSI biased to go high when DQSI is tri-state.
DATAVALID	O	Signal indicating the transmission of valid data to the FPGA core.
ECLKDQSR	O	Delay DQS used to capture the data.
DQCLK1	O	One clock edge, at the frequency of SCLK, used in output gearing.

Refer to the following technical note on the Lattice web site for more details:

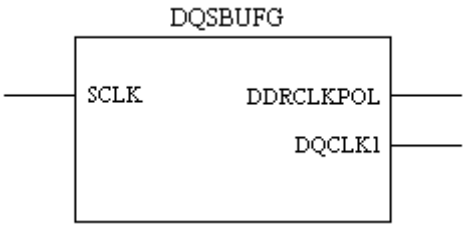
- TN1180 - LatticeECP3 High-Speed I/O Interface

DQSBUFG

DDR DQS Buffer Used for DDR_GENX1

Architectures Supported:

- LatticeECP3



INPUTS: SCLK

OUTPUT: DDRCLKPOL, DQCLK1

Description

DQSBUFG is the DDR DQS buffer used for DDR_GENX1 (DDR generic mode in X1 gearing).

- E: DDR_GENX1 (left/right/top)
- EA: Not allowed because it is not required

The table below describes the I/O ports.

Table 637:

Signal	I/O	Description
SCLK	I	System clock.
DDRCLKPOL	O	DDR clock polarity signal.
DQCLK1	O	One clock edge, at half the frequency of SCLK, used in output gearing.

Refer to the following technical note on the Lattice web site for more details:

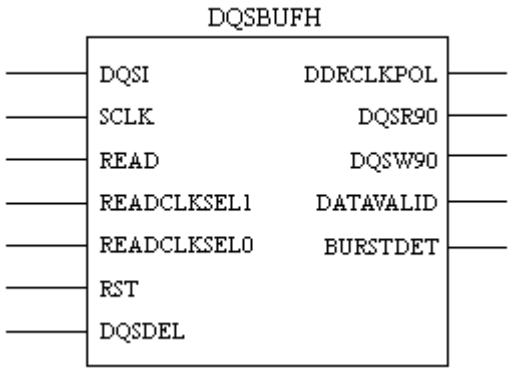
- TN1180 - LatticeECP3 High-Speed I/O Interface

DQSBUFH

DQS Circuit for DDR Memory

Architectures Supported:

- MachXO2
- Platform Manager 2



INPUTS: DQSI, SCLK, READ, READCLKSEL1, READCLKSEL0, RST, DQSDEL

OUTPUTS: DDRCLKPOL, DQSR90, DQSW90, DATAVALID, BURSTDET

ATTRIBUTES:

DQS_LI_DEL_ADJ: "PLUS" (default), "MINUS"

DQS_LI_DEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

DQS_LO_DEL_ADJ: "PLUS" (default), "MINUS"

DQS_LO_DEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

LPDDR: "DISABLED" (default), "ENABLED"

GSR: "ENABLED" (default), "DISABLED"

Description

DQSBUFH is the DQS circuit for DDR memory. It generates the 90 degree shift for DQS, and the DDRCLKPOL signal. It is used for right side only. The table below describes the I/O ports.

Table 638:

Signal	I/O	Description
DQSI	I	DQS signal from PIO.
READ	I	Signal for DDR read mode, coming from soft IP.
READCLKSEL1, READCLKSEL0	I	Select read clock source and polarity control for READ pulse position control in T/4 precision. The four positions are the rising/falling edges of SCLK or DQSW90. The signals come from soft IP.
SCLK	I	Clock from CIB.
RST	I	RESET for this block.
DQSDEL	I	DQS slave delay control from DQSDLLC.
DDRCLKPOL	O	SCLK polarity control.
DQSR90	O	DQS phase shifted by 90 degree output.
DQSW90	O	SCLK phase shifted by 90 degree output.
DATAVALID	O	Data valid signal for READ mode.
BURSTDET	O	Burst detection signal, moved from soft IP to hardware implementation.

Refer to the following technical note on the Lattice web site for more details:

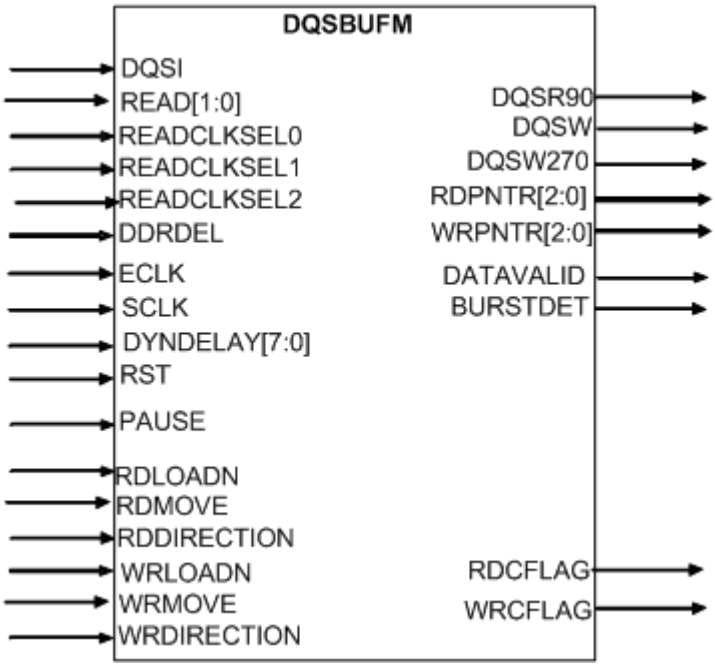
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

DQSBUFM

DQS Circuit for DDR Memory

Architectures Supported:

- ECP5



INPUTS: DQSI, SCLK, READ0, READ1, READCLKSEL0, READCLKSEL1, READCLKSEL2, DDRDEL, ECLK, SCLK, DYNDELAY0, DYNDELAY1, DYNDELAY2, DYNDELAY3, DYNDELAY4, DYNDELAY5, DYNDELAY6, DYNDELAY7, RST, PAUSE, RDLOADN, RDMOVE, RDDIRECTION, WRLOADN, WRMOVE, WRDIRECTION

OUTPUTS: DWSR90, DQSW, DQSW270, RDPNTR0, RDPNTR1, RDPNTR2, WRPNTR0, WRPNTR1, WRPNTR2, DATAVALID, BURSTDET, RDCFLAG, WRCFLAG

ATTRIBUTES:

DQS_LI_DEL_ADJ: "PLUS" (default), "MINUS"

DQS_LI_DEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

DQS_LO_DEL_ADJ: "PLUS" (default), "MINUS"

DQS_LO_DEL_VAL: integers 0~63 (PLUS), 1~64 (MINUS) (default: 0)

WRITE_LEVELING: Values: 0T, 1T (default: 0T)

Description

DQSBUFM element is used for DDR3 and DDR2 in X2 mode.

Table 639:

Port Name	I/O	Description
DQSI	I	DQS input from the DQS pin
DDRDEL	I	Delay code from DDRDLL
ECLK	I	Edge Clock
SCLK	I	System Clock
RST	I	Reset Input
READ[1:0]	I	Read input width for DQSBUFM
READCLKSEL0, READCLKSEL1, READCLKSEL2	I	Read clock pulse selection
DYNDELAY[7:0]	I	Dynamic Write leveling delay (only for DDR3)
PAUSE	I	Pause input to stop the DQSW/DQSW270 during write leveling or DDRDLL delay code change.
RDLOADN	I	Used to reset back to 90 degrees delay for Read Side DQS
RDMOVE	I	Pulse is required to change delay settings. The value on Direction will be sampled at “falling edge” of MOVE. Used to change delay on the Read side DQS.
RDDIRECTION	I	Indicates delay direction. “1” decrease delay count, “0” increase delay count. Used to change delay on the Read side DQS.
RDCFLAG	O	Indicates the delay counter has reached max value for the Read side DQS delay.
WRLOADN	I	Used to reset back to 90 degrees delay for Write Side DQS
WRMOVE	I	Pulse is required to change delay settings. The value on Direction will be sampled at “falling edge” of MOVE. Used to change delay on the Write side DQS.
WRDIRECTION	I	Indicates delay direction. “1” decrease delay count, “0” increase delay count. Used to change delay on the Write side DQS.
WRCFLAG	O	Indicates the delay counter has reached max value for the Write side DQS delay.
DQSR90	O	90 delay DQS used for Read
DQSW270	O	90 delay clock used for DQ Write
DQSW	O	Clock used for DQS Write
RDPNTR[2:0]	O	Read Pointer for IFIFO module
WRPNTR[2:0]	O	Write Pointer for IFIFO module

Table 639:

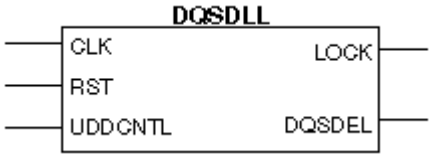
Port Name	I/O	Description
DATAVALID	O	Signal indicating start of valid data
BURSTDET	O	Burst Detect indicator

DQSDLL

DLL used as DDR memory DQS DLL

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeXP
- LatticeXP2



INPUTS: CLK, RST, UDDCNTL

OUTPUTS: LOCK, DQSDEL

ATTRIBUTES:

LOCK_SENSITIVITY: "LOW" (default), "HIGH"

Description

DQS delay calibration DLL. The primitive generates a 90-degree phase shift required for the DQS signal and implements the on-chip DQSDLL. Only one DQSDLL should be instantiated for all the DDR implementations on one half of the device. The clock input to this DLL should be at the same frequency as the DDR interface. DLL generates the delay based on this clock frequency and the update control input to this block. The DLL updates the dynamic delay

control to the DQS delay block when this update control (UDDCNTL) input is asserted. The active low signal on UDDCNTL updates the DQS phase alignment and should be initiated at the beginning of READ cycles.

Table 640:

Port Name	I/O	Definition
CLK	I	System CLK should be at the frequency of the DDR interface from the FPGA core.
RST	I	Resets the DQSDLL
UDDCNTL	I	This is an active low port. It provides an update signal to the DLL that will update the dynamic delay. When held low, this signal will update the DQSDEL.
LOCK	O	Indicates when the DLL is in phase.
DQSDEL	O	The digital delay generated by the DLL, should be connected to the DQSBUF primitive.

DQSDLL Configuration: By default, this DLL generates a 90-degree phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL. You can control the sensitivity to jitter by using the LOCK_SENSITIVITY attribute. This configuration bit can be programmed to be either “HIGH” or “LOW”. The DLL Lock Detect circuit has two modes of operation controlled by the LOCK_SENSITIVITY bit, which selects more or less sensitivity to jitter. If this DLL is operated at or above 150 MHz, it is recommended that the LOCK_SENSITIVITY bit be programmed “HIGH” (more sensitive). When running at or below 100 MHz, it is recommended that the bit be programmed “LOW” (more tolerant). For 133 MHz, the LOCK_SENSITIVITY bit can go either way.

Refer to the following technical notes on the Lattice web site for more details

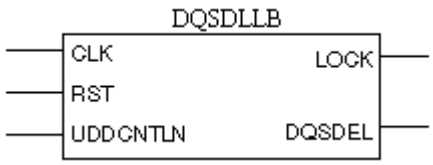
- TN1050 - LatticeECP/EC and LatticeXP DDR Usage Guide
- TN1105 - LatticeECP2/M High-Speed I/O Interface
- TN1138 - LatticeXP2 High-Speed I/O Interface

DQSDLLB

DQS DLL for DDR_MEM, DDR2_MEM, and DDR3_MEM

Architectures Supported:

- LatticeECP3



INPUTS: CLK, RST, UDDCNTLN

OUTPUTS: LOCK, DQSDEL

ATTRIBUTES:

LOCK_SENSITIVITY: "LOW" (default), "HIGH"

Description

DQSDLLB is the DLL used as DDR memory DQS DLL.

- E and EA: DDR_MEM, DDR2_MEM, and DDR3_MEM (left/right)

The table below describes the I/O ports.

Table 641:

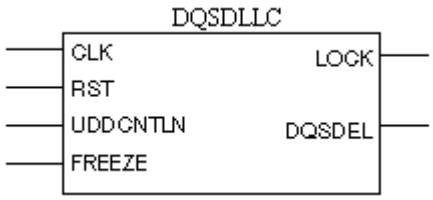
Signal	I/O	Description
CLK	I	Clock from the CIB, coming from a PLL. The clock should run at the DDR memory frequency.
RST	I	RESET input to the master DLL.
UDDCNTLN	I	Update control generated from the core.
DQSDEL	O	DQS delay generated by DQSDLL.
LOCK	O	Lock output of the DQSDLL to the CIB.

DQSDLLC

Master DLL for Generating Required Delay

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLK, RST, UDDCNTLN, FREEZE

OUTPUTS: LOCK, DQSDEL

ATTRIBUTES:

DEL_ADJ: "PLUS" (default), MINUS

DEL_VAL: integers 0-127 (PLUS), 1-128 (MINUS) (default: 0)

LOCK_SENSITIVITY: "LOW" (default), "HIGH"

FIN: value range supported by DLL (default: "100.0")

FORCE_MAX_DELAY: "NO" (default), "YES"

GSR: "ENABLED" (default), "DISABLED"

Description

DQSDLLC is the master DLL to generate required delay. See the table below for I/O port descriptions.

Table 642:

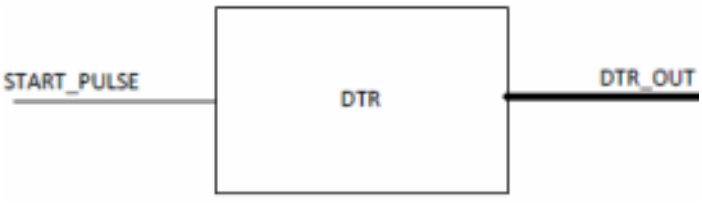
Signal	I/O	Description
CLK	I	Clock from the CIB.
RST	I	DLL reset control.
UDDCNTLN	I	Hold/update control to delay code before adjustment.
FREEZE	I	Signal used to freeze or release DLL input CLK.
DQSDEL	O	DLL delay control code to slave delay cells. Connected to DQSDEL of the DQSBUFH element.
LOCK	O	DLL lock signal.

DTR

Digital Temperature Readout

Architectures Supported:

- ECP5



INPUTS: START_PULSE

OUTPUTS: DTR_OUT

Description

Digital Temperature Readout (or DTR) is an on board temperature sensing circuit that provides the junction temperature of the die while running.

See the table below for I/O port descriptions.

Table 643:

Signal	I/O	Description
START_PULSE	I	Pulse to instruct DTR to start capturing temperature.
DTR_OUT	O	8-bit DTR output

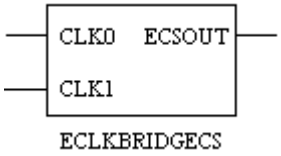
E

ECLKBRIDGECS

ECLK Bridge Block Clock Select

Architectures Supported:

- ECP5
- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLK0, CLKI, SEL

OUTPUT: ECSOUT

Description

ECLK high speed bridge eases the design of high speed video or DDRX4 mode applications. It takes the high speed edge clocks sources from both sides (top and bottom). The bridge enhances the communication of ECLKs across the die. The bridge is supported for devices equal to or above 1200 LUTs.

The table below describes the I/O ports of the ECLKBRIDGECS primitive.

Table 644:

Port	I/O	Unused Port	Function
CLK0	I	Tie low	Input clock to the edge bridge clock select.
CLK1	I	Tie low	Input clock to the edge bridge clock select.
SEL	I	Tie low	From CIB, edge bridge clock select.
ECSOUT	O	Dangle	Output from edge bridge clock select.

ECLKBRIDGECS Usage with VHDL

Library Instantiation

```
library lattice;
use lattice.components.all;
```

Component Declaration

```
component ECLKBRIDGECS
   port (CLK0   : in std_logic;
         CLK1   : in std_logic;
         SEL    : in std_logic;
         ECSOUT : out std_logic);
end component;
```

ECLKBRIDGECS Instantiation

```
I1: ECLKBRIDGECS
   port map (CLK0   => CLK0;
             CLK1   => CLK1;
             SEL    => SEL;
             ECSOUT => ECSOUT);
end component;
```

ECLKBRIDGECS Usage with Verilog HDL

Component Declaration

```
module ECLKBRIDGECS (CLK0,
                     CLK1,
                     SEL,
                     ECSOUT);
input CLK0;
input CLK1;
input SEL;
output ECSOUT;
endmodule
```

ECLKBRIDGECS Instantiation

```
ECLKBRIDGECS I1 (.CLK0 (CLK0),
                 .CLK1 (CLK1),
                 .SEL   (SEL),
                 .ECSOUT (ECSOUT));
```

For more information and usage, refer to the following technical note on the Lattice web site.

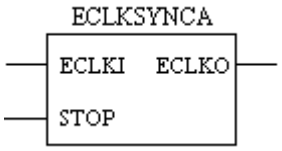
- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide

ECLKSYNCA

ECLK Stop Block

Architectures Supported:

- LatticeECP3
- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: ECLKI, STOP

OUTPUT: ECLKO

Description

ECLKSYNCA is the optional ECLK synchronization for DDR_MEM (DDR memory mode), DDR2_MEM (DDR2 memory mode), and DDR3_MEM (DDR3 memory mode).

- E and EA: DDR_MEM, DDR2_MEM, and DDR3_MEM (left/right/top)

The table below describes the I/O ports of the ECLKSYNCA primitive.

Table 645:

Port	I/O	Unused Port	Function
ECLKI	I	Tie low	Edge clock input to the stop clock
STOP	I	Tie low	Control signal to stop the edge clock to synchronize the signals derived from ECLK
ECLKO	O	Dangle	Edge clock output from the stop clock

ECLKSYNCA Usage with VHDL

Library Instantiation

```
library lattice;
use lattice.components.all;
```

Component Declaration

```
component ECLKSYNCA
   port (ECLKI : in std_logic;
         STOP  : in std_logic;
         ECLKO : out std_logic);
end component;
```

ECLKSYNCA Instantiation

```
I1: ECLKSYNCA
   port map (ECLKI => ECLKI,
             STOP => STOP,
             ECLKO => ECLKO);
end component;
```

ECLKSYNCA Usage with Verilog HDL

Component Declaration

```
module ECLKSYNCA (ECLKI,  STOP, ECLKO);
input ECLKI;
input STOP;
output ECLKO;
endmodule
```

ECLKSYNCA Instantiation

```
ECLKSYNCA I1 (.ECLKI (ECLKI);
              .STOP (STOP);
              .ECLKO (ECLKO));
```

For more information and usage, refer to the following technical notes on the Lattice web site.

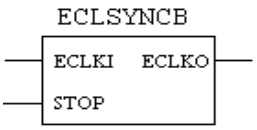
- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide
- TN1177 - LatticeECP3 sysIO Usage Guide

ECLKSYNCB

ECLK Stop Block

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: ECLKI, STOP

OUTPUT: ECLKO

Description

ECLKSYNCB is the optional ECLK synchronization for DDR_MEM (DDR memory mode), DDR2_MEM (DDR2 memory mode), and DDR3_MEM (DDR3 memory mode). The ECP5 ECLKSYNC is similar to MachXO2 and LaticeECP3 (ECLKSYNA), however it has some added latency.

Asserting the STOP control signal (from CIB) will stop the edge clock and synchronize the signals from ECLK used in high speed DDR mode applications such as DDR memory, generic DDR and 7:1 LVDS. The STOP signal is synchronized with ECLK when asserted. When the STOP signal is released, every clock toggling from the second rising edge clock & after will be output.

The table below describes the I/O ports of the ECLKSYNCB primitive.

Table 646:

Port	I/O	Unused Port	Function
ECLKI	I	Tie low	Edge clock input to the stop block
STOP	I	Tie low	Control signal to stop the edge clock
ECLKO	O	Dangle	Edge clock output from the stop block

ECLKSYNCB Usage with VHDL

Component Declaration

```
component ECLKSYNCB port (
ECLKI : in std_logic;
STOP : in std_logic;
ECLKO : out std_logic);
end component;
```

ECLKSYNCB Instantiation

```
I1: ECLKSYNCB port map (
ECLKI => ECLKI,
STOP => STOP,
ECLKO => ECLKO);
end component;
```

ECLKSYNCB Usage with Verilog HDL

Component Declaration

```
module ECLKSYNCB (ECLKI, STOP, ECLKO)
input ECLKI;
input STOP;
output ECLKO;
endmodule
```

ECLKSYNCB Instantiation

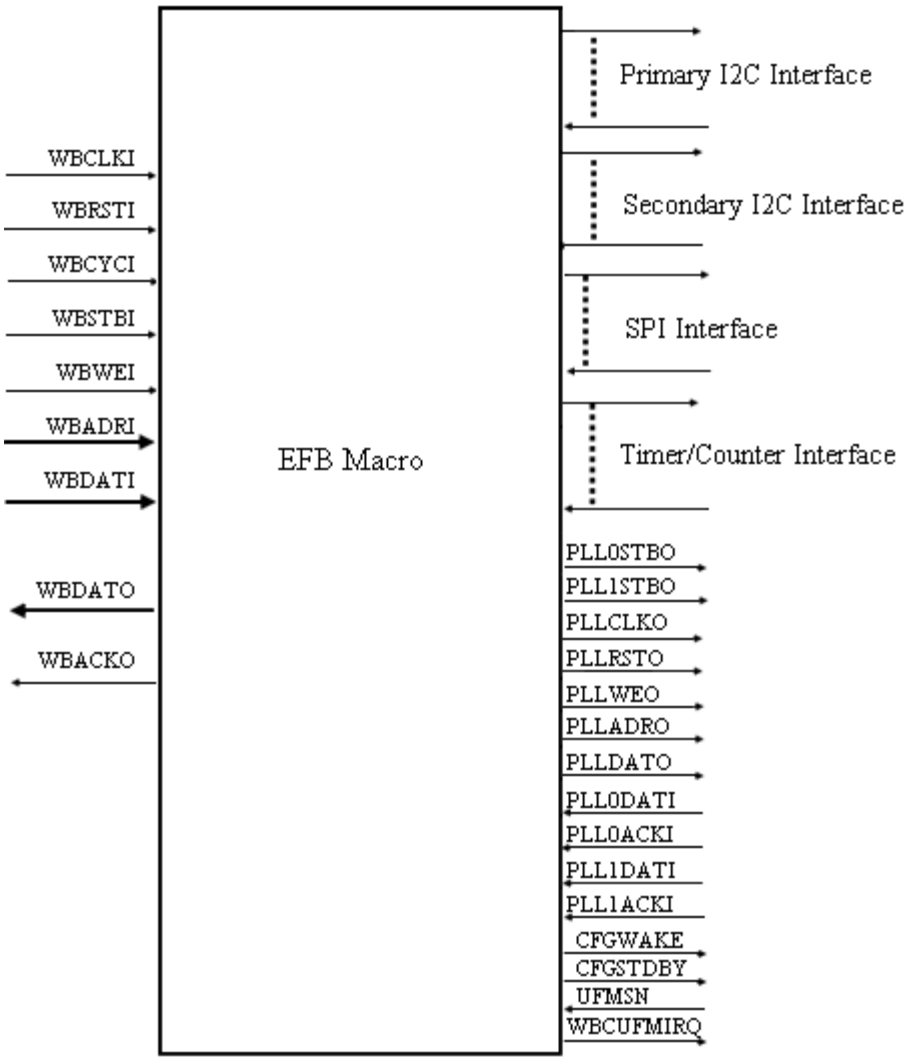
```
ECLKSYNCB I1 (.ECLKI (ECLKI), .STOP (STOP), .ECLKO (ECLKO));
```

EFB

Embedded Function Block

Architectures Supported:

- MachXO2
- MachXO3L
- Platform Manager 2



INPUTS: WBCLKI, WBRSTI, WBCYCI, WBSTBI, WBWEI, WBADRI7, WBADRI6, WBADRI5, WBADRI4, WBADRI3, WBADRI2, WBADRI1, WBADRI0, WBDATI7, WBDATI6, WBDATI5, WBDATI4, WBDATI3, WBDATI2, WBDATI1, WBDATI0, PLL0DATI7, PLL0DATI6, PLL0DATI5, PLL0DATI4, PLL0DATI3, PLL0DATI2, PLL0DATI1, PLL0DATI0, PLL0ACKI, PLL1DATI7, PLL1DATI6, PLL1DATI5, PLL1DATI4, PLL1DATI3, PLL1DATI2, PLL1DATI1, PLL1DATI0, PLL1ACKI, I2C1SCLI, I2C1SDAI, I2C2SCLI, I2C2SDAI, SPISCKI, SPIMISOI, SPIMOSII, SPISCSN, TCCLKI, TCRSTN, TCIC, UFMSN

OUTPUTS: WBDATO7, WBDATO6, WBDATO5, WBDATO4, WBDATO3, WBDATO2, WBDATO1, WBDATO0, WBACKO, PLLCLKO, PLLRSTO, PLL0STBO, PLL1STBO, PLLWEO, PLLADRO4, PLLADRO3, PLLADRO2, PLLADRO1, PLLADRO0, PLLDATO7, PLLDATO6, PLLDATO5, PLLDATO4,

PLLDATO3, PLLDATO2, PLLDATO1, PLLDATO0, I2C1SCLO, I2C1SCLOEN, I2C1SDAO, I2C1SDAOEN, I2C2SCLO, I2C2SCLOEN, I2C2SDAO, I2C2SDAOEN, I2C1IRQO, I2C2IRQO, SPISCKO, SPISCKEN, SPIMISOO, SPIMISOEN, SPIMOSIO, SPIMOSIEN, SPIMCSN0, SPIMCSN1, SPIMCSN2, SPIMCSN3, SPIMCSN4, SPIMCSN5, SPIMCSN6, SPIMCSN7, SPICSNEN, SPIIRQO, TCINT, TCOC, WBCUFMIRQ, CFGWAKE, CFGSTDBY

Description

The EFB primitive has seven explicit interfaces: WISHBONE, SPI, I2C (Primary), I2C (Secondary), Timer/Counter, and two PLLs.

DEV_DENSITY:

MachXO2: "256L", "640L", "1200L", "2000L", "4000L", "7000L", "10000L", "640U", "1200U", "2000U", "4000U"

MachXO3LF: "640L_121P", "1300L", "2100L", "4300L", "1300L_256P", "2100L_324P", "4300L_400P", "6900L"

For detailed information regarding the GUI, interface, and usage regarding each EFB interface, refer to the following technical note on the Lattice web site:

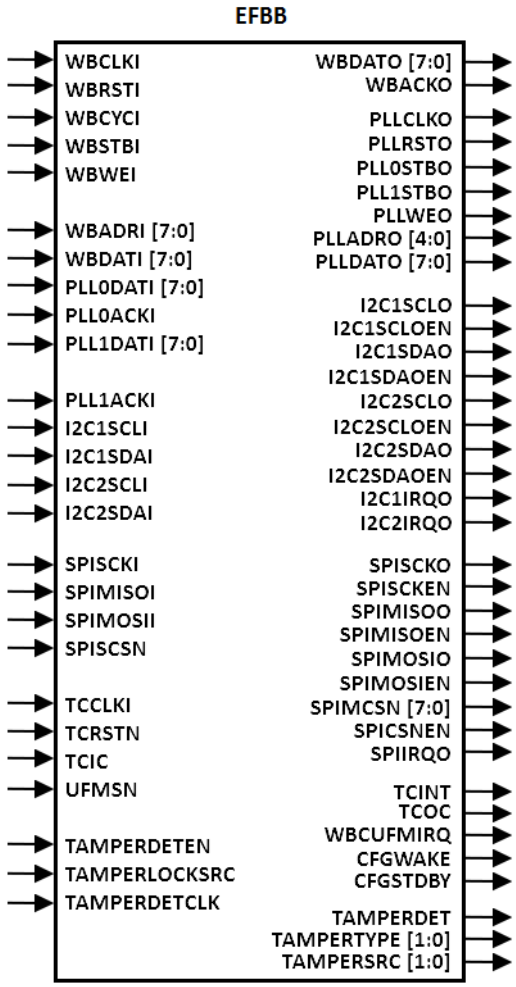
- TN1205 - Using User Flash Memory and Hardened Control Functions in MachXO2 Devices

EFBB

Embedded Function Block for MachXO3D

Architectures Supported:

- LFMNX
- MachXO3D



INPUTS: WBCLKI, WBRSTI, WBCYCI, WBSTBI, WBWEI, WBADRI7, WBADRI6, WBADRI5, WBADRI4, WBADRI3, WBADRI2, WBADRI1, WBADRI0, WBDATI7, WBDATI6, WBDATI5, WBDATI4, WBDATI3, WBDATI2, WBDATI1, WBDATI0, PLL0DATI7, PLL0DATI6, PLL0DATI5, PLL0DATI4, PLL0DATI3, PLL0DATI2, PLL0DATI1, PLL0DATI0, PLL0ACKI, PLL1DATI7, PLL1DATI6, PLL1DATI5, PLL1DATI4, PLL1DATI3, PLL1DATI2, PLL1DATI1, PLL1DATI0, PLL1ACKI, I2C1SCLI, I2C1SDAI, I2C2SCLI, I2C2SDAI, SPISCKI, SPIMISOI, SPIMOSII, SPISCSN, TCCLKI, TCRSTN, TCIC, UFMSN,

OUTPUTS: WBDATO7, WBDATO6, WBDATO5, WBDATO4, WBDATO3, WBDATO2, WBDATO1, WBDATO0, WBACKO, PLLCLKO, PLLRSTO, PLL0STBO, PLL1STBO, PLLWEO, PLLADRO4, PLLADRO3, PLLADRO2, PLLADRO1, PLLADRO0, PLLDATO7, PLLDATO6, PLLDATO5, PLLDATO4, PLLDATO3, PLLDATO2, PLLDATO1, PLLDATO0, I2C1SCLO, I2C1SCLOEN, I2C1SDAO, I2C1SDAOEN, I2C2SCLO, I2C2SCLOEN, I2C2SDAO, I2C2SDAOEN, I2C1IRQO, I2C2IRQO,SPISCKO, SPISCKEN, SPIMISOO, SPIMISOEN, SPIMOSIO, SPIMOSIEN,SPIMCSN0, SPIMCSN1, SPIMCSN2, SPIMCSN3, SPIMCSN4, SPIMCSN5, SPIMCSN6, SPIMCSN7, SPICSNEN, SPIIRQO, TCINT, TCOC, WBCUFMIRQ, CFGWAKE, CFGSTDBY;

Description:

The EFBB primitive is based on EFB primitive with addition of parameters related to UFM.

ATTRIBUTES:

EFB_I2C1: “DISABLED”

EFB_I2C2: “DISABLED”

EFB_SPI: “DISABLED” (default)

EFB_TC: “DISABLED” (default)

EFB_TC_PORTMODE: “NO_WB”, DISABLED” (default)

EFM_UFM_BOOT: “INT_SINGLE_BOOT_CFG0” (default);
INT_SINGLE_BOOT_CFG0_UFM0, NT_DUAL_BOOT_CFG0_CFG1,
INT_DUAL_BOOT_CFG0_UFM0_CFG1,
INT_DUAL_BOOT_CFG0_CFG1_UFM1,
INT_DUAL_BOOT_CFG0_UFM0_CFG1_UFM1, EXTERNAL_BOOT

EFB_UFM: “DISABLED” (default); “ENABLED”

EFB_UFM0: “DISABLED”

EFB_UFM1: “DISABLED”

EFB_UFM2: “DISABLED”

EFB_UFM3: “DISABLED”

EFB_CFG0: “DISABLED”

EFB_CFG1: “DISABLED”

EFB_WB_CLK_FREQ: “50” (default)

DEV_DENSITY: “9400L” (default)

UFM0_INIT_PAGES: “0”

UFM0_INIT_START_PAGE: “0” (default)

UFM0_INIT_ALL_ZEROS: “ENABLED” (default)

UFM0_INIT_FILE_NAME: “None” (default)

UFM0_INIT_FILE_FORMAT: “Hex” (default)

UFM1_INIT_PAGES: “0” (default)

UFM1_INIT_START_PAGE: “0” (default)

UFM1_INIT_ALL_ZEROS: “ENABLED” (default)

UFM1_INIT_FILE_NAME: “None” (default)

UFM1_INIT_FILE_FORMAT: “Hex” (default)

UFM2_INIT_PAGES: “0” (default)

UFM2_INIT_START_PAGE: “0” (default)

UFM2_INIT_ALL_ZEROS: “ENABLED” (default)

UFM2_INIT_FILE_NAME: “None” (default)

UFM2_INIT_FILE_FORMAT: “Hex” (default)

UFM3_INIT_PAGES: “0” (default)

UFM3_INIT_START_PAGE: “0” (default)

UFM3_INIT_ALL_ZEROS: “ENABLED” (default)

UFM3_INIT_FILE_NAME: “None” (default)

UFM3_INIT_FILE_FORMAT: “Hex” (default)

CFG0_INIT_PAGES: “0” (default)

CFG0_INIT_START_PAGE: “0” (default)

CFG0_INIT_ALL_ZEROS: “ENABLED” (default)

CFG0_INIT_FILE_NAME: “None” (default)

CFG0_INIT_FILE_FORMAT: “Hex” (default)

CFG1_INIT_PAGES: “0” (default)

CFG1_INIT_START_PAGE: “0” (default)

CFG1_INIT_ALL_ZEROS: “ENABLED” (default)

CFG1_INIT_FILE_NAME: “None” (default)

CFG1_INIT_FILE_FORMAT: “Hex” (default)

I2C1_ADDRESSING: “7BIT”

I2C1_SLAVE_ADDR: “0b10000001”

I2C1_BUS_PERF: “100kHz”

I2C1_CLK_DIVIDER: “1”

I2C1_GEN_CALL: “DISABLED”

I2C1_WAKEUP: “DISABLED”

I2C2_ADDRESSING: “7BIT”

I2C2_SLAVE_ADDR: “0b10000001”

I2C2_BUS_PERF: “100kHz”

I2C2_CLK_DIVIDER: “1”

I2C2_GEN_CALL: “DISABLED”

I2C2_WAKEUP: “DISABLED”

SPI_MODE: “SLAVE”

SPI_CLK_DIVIDER: “1”

SPI_LSB_FIRST: “DISABLED”

SPI_CLK_INV: “DISABLED”

SPI_PHASE_ADJ: “DISABLED”

SPI_SLAVE_HANDSHAKE: “DISABLED”

SPI_INTR_TXRDY: “DISABLED”

SPI_INTR_RXRDY: “DISABLED”

SPI_INTR_TXOVR: “DISABLED”

SPI_INTR_RXOVR: “DISABLED”

SPI_WAKEUP: “DISABLED”

TC_MODE: “CTCM”

TC_CCLK_SEL: “1”

TC_SCLK_SEL: “PCLOCK”

GSR: “ENABLED”

TC_OCR_SET: “32767”

TC_OC_MODE: “TOGGLE”

TC_RESETN: “ENABLED”

TC_TOP_SET: “65535”

TC_TOP_SEL: “ON”

TC_OV_INT: “OFF”

TC_OCR_INT: “OFF”

TC_ICR_INT: “OFF”

TC_OVERFLOW: “ENABLED”

TC_ICAPTURE: “DISABLED’

EFB_TAMPER_TYPE_PASSWORD: “DISABLED”

EFB_TAMPER_TYPE_LOCKED_FLASH_SRAM: “DISABLED”

EFB_TAMPER_TYPE_MANUFACTURE_MODE: “DISABLED”

EFB_TAMPER_SRC_JTAG: “DISABLED”

EFB_TAMPER_SRC_SSPI: “DISABLED”

EFB_TAMPER_SRC_SI2C: “DISABLED”

EFB_TAMPER_SRC_WB: “DISABLED”

EFB_TAMPER_PORT_LOCK: “DISABLED”

EFB_TAMPER_DETECTION_RESPONSE: “DISABLED”

Description:

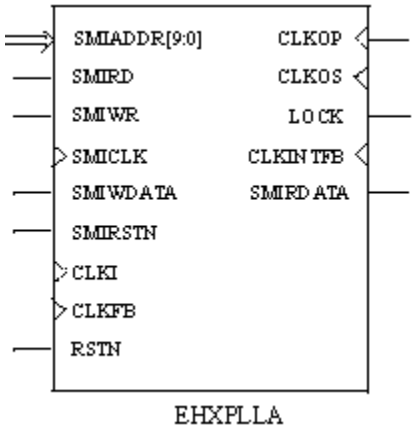
Embedded Function Block (EFB) (formerly known as the CFG block). The EFB includes a SPI, two I2C’s, and a timer/counter peripheral.

EHXPLLA

Enhanced High Performance with Dynamic Input Delay Control PLL

Architectures Supported:

- LatticeSC/M



INPUTS: SMIADDR9, SMIADDR8, SMIADDR7, SMIADDR6, SMIADDR5, SMIADDR4, SMIADDR3, SMIADDR2, SMIADDR1, SMIADDR0, SMIRD, SMIWR, SMICLK, SMIWDATA, SMIRSTN, CLKI, CLKFB, RSTN

OUTPUTS: CLKOP, CLKOS, LOCK, CLKINTFB, SMIRDATA

ATTRIBUTES:

CLKI_DIV: integers 1~64 (default: 1)

CLKFB_DIV: integers 1~64 (default: 1)

CLKOP_DIV: integers 1~64 (default: 1)

CLKOS_DIV: integers 1~64 (default: 1)

CLKI_FDEL: 0 (default), 100, 200, ..., 700

CLKFB_FDEL: 0 (default), 100, 200, ..., 700

CLKOS_FDEL: 0 (default), 100, 200, ..., 700

CLKOP_MODE: "BYPASS" (default), "FDEL0", "VCO", "DIV"

CLKOS_MODE: "BYPASS" (default), "FDEL", "VCO", "DIV"

PHASEADJ: 0 (default), 45, 90, 135, 190, 225, 270, 315

GSR: "ENABLED" (default), "DISABLED"

SMI_OFFSET: 0x400~0x7FF (default: 12'h410)

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

CLKOS_VCODEL: integers 0~31 (default: 0)

MODULE_TYPE: "EHXPLLA"

IP_TYPE: "EHXPLLA"

Description

The Enhanced Extended Performance PLL (EHXPLLA) includes all features available in the PLL. This primitive includes SMI access so that you may configure the PLL as you require. The EHXPLLA primitive can be created through IPexpress. Note that Some combination of legal values are not allowed, due to other system limitations, such as the frequency of operation.

The following are descriptions of EHXPLLA port functions.

Table 647:

Port	I/O	Function
CLKI	I	CLKI[1:3]: from CIBs CLKI[0]: dedicated clock input pin Frequency: 2~1000 MHz
CLKFB	I	CLKFB[2,3]: from CIBs CLKFB[1]: dedicated external feedback pin CLKFB[0]: internal feedback from VCO output (CLKINTFB) Frequency: 2~1000 MHz
CLKOP	O	PLL output clock – main clock output Frequency: 1.5625~1000 MHz
CLKOS	O	PLL output clock – supplemental clock output Frequency: 1.5625~1000 MHz
LOCK	O	PLL locked to CLK1
CLKINTFB	O	CLKFB internal feedback source from VCO output
RSTN	I	Active low reset
SMIADDR[9:0]	I	SMI address bus
SMICLK	I	SMI clock signal
SMIRSTN	I	SMI reset signal
SMIRD	I	SMI read signal
SMIWDATA	I	SMI write data input
SMIWR	I	SMI write signal
SMIRDATA	O	SMI read data output

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

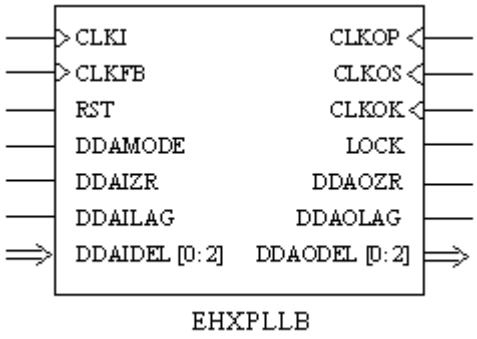
- TN1098 - LatticeSC sysCLOCK PLL/DLL User's Guide

EHXPLLB

Enhanced High Performance with Dynamic Input Delay Control PLL

Architectures Supported:

- LatticeECP/EC
- LatticeXP



EHXPLLB

INPUTS: CLKI, CLKFB, RST, DDAMODE, DDAIZR, DDAILAG, DDAIDEL0, DDAIDEL1, DDAIDEL2

OUTPUTS: CLKOP, CLKOS, CLKOK, LOCK, DDAOZR, DDAOLAG, DDAODEL0, DDAODEL1, DDAODEL2

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKI_DIV: (*LatticeECP/EC*) integers 1~16 (default: 1);
(*LatticeXP*) integers 1~15 (default: 1)

CLKFB_DIV: (*LatticeECP/EC*) integers 1~16 (default: 1);
(*LatticeXP*) integers 1~15 (default: 1)

CLKOP_DIV: (*LatticeECP/EC*) even integers 2~32 if CLKOS is not used; 2, 4, 8, 16, 32 is CLKOS is used. (Default: 8)
(*LatticeXP*) even integers 2~30 if CLKOS is not used (default: 6); 2, 4, 8, 16 if CLKOS is used (default: 4).

CLKOK_DIV: 2 (default), 4, 6, 8, ..., 126, 128

FDEL: integers -8~8 (default: 0)

PHASEADJ: 0 (default), 45, 90, 135, 190, 225, 270, 315

DUTY: integers 1~7 (default: 4)

DELAY_CNTL: "STATIC" (default), "DYNAMIC"

Description

The following are descriptions of EHXPLLB port functions.

Table 648:

Port	I/O	Function
CLKI	I	Global clock input; frequency: 20~420 MHz.
CLKFB	I	External feedback, internal feedback from CLKOP divider; frequency: 20~420 MHz.
RST	I	"1" to reset M-divider.
DDAMODE	I	DDA mode. "1": pin control (dynamic); "0": fuse control (static).
DDAIZR	I	DDA delay zero. "1": delay = 0; "0": delay = on.
DDAILAG	I	DDA lag/lead. "1": lag; "0": lead.
DDAIDEL[0:2]	I	DDA delay.
CLKOP	O	PLL output clock to clock tree (no phase shift); frequency: 20~420 MHz.
CLKOS	O	PLL output clock to clock tree (phase shifted/duty cycle changed); frequency: 20~420 MHz.
CLKOK	O	PLL output to clock tree (K divider, low speed, output); frequency: 0.156~210 MHz.
LOCK	O	"1" indicates PLL LOCK to CLK_IN.
DDAOZR	O	DDA delay zero output.
DDAOLAG	O	DDA lag/lead output.
DDAODEL[0:2]	O	DDA delay output.

Dynamic Delay Adjustment

The Dynamic Delay Adjustment is controlled by DDAMODE input. This feature is available in EHXPLLB primitive only. When the DDAMODE input is set to “1,” the delay control is done through the inputs, DDAIZR, DDAILAG and DDAIDEL(2:0). For this mode, the attribute “DELAY_CNTL” must be set to “DYNAMIC.”

Equations for Generating Input and Output Frequency Ranges

These values of fIN, fOUT, fVCO are the absolute frequency ranges for the PLL. The values of fINMIN, fINMAX, fOUTMIN, and fOUTMAX, are the calculated frequency ranges based on the divider settings. These calculated frequency ranges become the limits for the specific divider settings used in the design.

fOUT = fIN * (N/M)

fVCO = fOUT * V = fIN * (N/M) * V

fIN = (fVCO /(V*N))*M

Where M = CLKI DIV

N = CLKFB DIV

V = CLKOP DIV

K = CLKOK DIV

fINMIN = ((fVCOMIN /(V*N))*M, if below 33 * M round up to 33 * M

fINMAX = (fVCOMAX/(V*N))*M, if above 420 round down to 420

fOUTMIN = fINMIN*(N/M), if below 33 * N round up to 33 * N

fOUTMAX = fINMAX*(N/M), if above 420 round down to 420

fOUTKMIN = fOUTMIN /K

fOUTKMAX = fOUTMAX /K

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

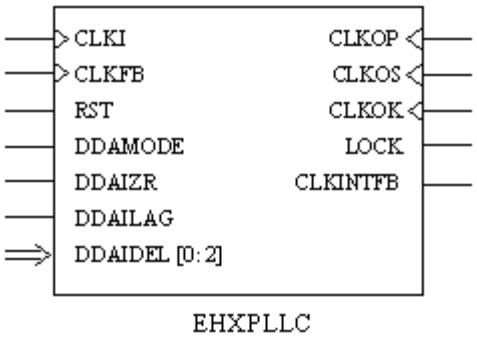
- TN1049 - LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide

EHXPLLC

Enhanced Extended Performance PLL

Architectures Supported:

- MachXO
- Platform Manager



INPUTS: CLKI, CLKFB, RST, DDAMODE, DDAIZR, DDAILAG, DDAIDEL0, DDAIDEL1, DDAIDEL2

OUTPUTS: CLKOP, CLKOS, CLKOK, LOCK, CLKINTFB

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKFB_DIV: integers 1~16 (default: 1)

CLKI_DIV: integers 1~16 (default: 1)

CLKOP_DIV: even integers 2~32 if CLKOS is not used; 2, 4, 8, 16, 32 if CLKOS is used. (Default: 8)

CLKOK_DIV: 2 (default), 4, 6, 8, ..., 126, 128

DELAY_CNTL: "STATIC" (default), "DYNAMIC"

FDEL: integers -8~8 (default: 0)

PHASEADJ: 0 (default), 45, 90, 135, 190, 225, 270, 315

DUTY: integers 1~7 (default: 4)

Description

The EHXPLLC primitive is used for MachXO and Platform Manager PLL implementation. The definitions of the PLL I/O ports are shown in the following table. The EHXPLLC includes all features available in the MachXO or Platform Manager PLL.

Table 649:

Port	I/O	Function
CLKI	I	General routing or dedicated global clock input pad.
CLKFB	I	From general routing, clock tree, internal feedback from CLKOP or dedicated external feedback CLKFB Ipad.
RST	I	"1" to reset PLL counters.
CLKOP	O	PLL output clock to clock tree (no phase shift).
CLKOS	O	PLL output clock to clock tree (phase shifted/duty cycle changed).
CLKOK	O	PLL output to clock tree (CLKOK divider, low speed, output).
LOCK	O	"1" indicates PLL LOCK to CLKI; asynchronous signal.
CLKINTFB	O	Internal feedback source. CLKOP divider output before CLOCK TREE.
DDAMODE	I	DDA mode. "1": pin control (dynamic); "0": fuse control (static).
DDAIZR	I	DDA delay zero. "1": delay = 0; "0": delay = on.

Table 649:

Port	I/O	Function
DDAILAG	I	DDA lag/lead. "1": lead; "0": lag.
DDAIDEL[0:2]	I	DDA delay.

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

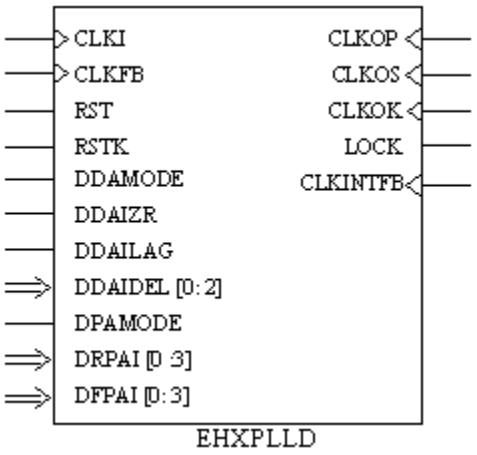
- TN1089 - MachXO sysCLOCK Design and Usage Guide

EHXPLLD

Complex PLL

Architectures Supported:

- LatticeECP2/M



INPUTS: CLKI, CLKFB, RST, RSTK, DDAMODE, DDAIZR, DDAILAG, DDAIDEL0, DDAIDEL1, DDAIDEL2, DPAMODE, DRPAI3, DRPAI2, DRPAI1, DRPAI0, DFPAI3, DFPAI2, DFPAI1, DFPAI0

OUTPUTS: CLKOP, CLKOK, CLKOS, LOCK, CLKINTFB

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKI_DIV: integers 1~64 (default: 1)

CLKFB_DIV: integers 1~16 (default: 1)

CLKOP_DIV: 2, 4, 8 (default), 16, 32, 48, 64, 80, 96, 112, 128

CLKOK_DIV: 2 (default), 4, 6, 8, ..., 126, 128

FDEL: integers -8~8 (default: 0)

PHASEADJ: 0 (default), 22.5, 45, 67.5, 90, ..., 315, 337.5

DUTY: integers 2~14 (default: 8)

PHASE_CNTL: "STATIC" (default), "DYNAMIC"

DELAY_CNTL: "STATIC" (default), "DYNAMIC"

PLLCAP: "DISABLED" (default), "ENABLED", "AUTO"

CLKOP_BYPASS: "DISABLED" (default), "ENABLED"

CLKOS_BYPASS: "DISABLED" (default), "ENABLED"

CLKOK_BYPASS: "DISABLED" (default), "ENABLED"

Description

The ECP2 devices provide two type of PLLS: SPLL and GPLL. The SPLL is a baseline PLL. The GPLL includes all features of SPLL plus Dynamic Delay Adjustment. The primitive for the SPLL is EPLLD. The primitive for the GPLL are EPLLD and EHXPLLD. See the following table for GPLL and SPLL IO port description.

Table 650:

Port	I/O	Function
CLKI	I	Input clock.
CLKFB	I	Feedback clock.
RST	I	PLL reset (connect to CNTRST port). High active reset.
RSTK	I	Reset for K divider (connect to RESETK port). High active reset.
CLKOP	O	PLL output clock (no phase shift).
CLKOS	O	PLL output clock (phase shifted/duty cycle changed).
CLKOK	O	PLL output to clock tree (no phase shift, low speed).
LOCK	O	PLL LOCK to CLKI, asynchronous signal. Active high indicates PLL lock.
CLKINTFB	O	Internal feedback source. CLKOP divider output before CLOCK TREE.
DDAMODE	I	DDA mode. Active high indicates pin control (DYNAMIC) and active low indicates fuse control (STATIC).
DDAIZR	I	DDA delay zero. Active high indicates “delay = 0” and active low indicates “delay= on.”

Table 650:

Port	I/O	Function
DDAILAG	I	DDA lag/lead. Active high indicates “lead” and active low indicates “lag.”
DDAIDEL[2:0]	I	DDA delay.
DPAMODE	I	Dynamic phase adjust mode. Active high indicates pin control (DYNAMIC) and active low indicates fuse control (STATIC).
DRPAI[3:0]	I	Dynamic coarse phase shift, rising edge setting.
DFPAI[3:0]	I	Dynamic coarse phase shift, falling edge setting.

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

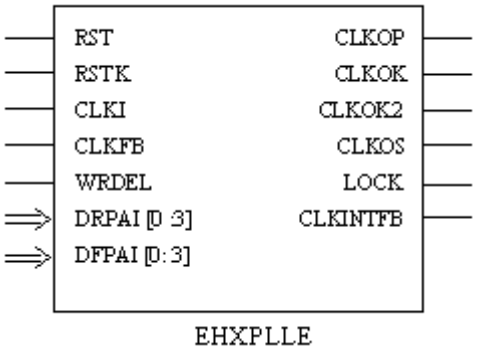
- TN1103 - LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide

EHXPLLE

Complex PLL

Architectures Supported:

- LatticeXP2



INPUTS: CLKI, CLKFB, RST, RSTK, WRDEL, DRPAI3, DRPAI2, DRPAI1, DRPAI0, DFPAI3, DFPAI2, DFPAI1, DFPAI0

OUTPUTS: CLKOP, CLKOK, CLKOK2, CLKOS, LOCK, CLKINTFB

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKFB_DIV: integers 1~64 (default: 1)

CLKI_DIV: integers 1~64 (default: 1)

CLKOP_DIV: 2, 4, 8 (default), 16, 32, 48, 64, 80, 96, 112, 128

CLKOK_DIV: 2 (default), 4, 6, ..., 126, 128

PHASE_CNTL: "STATIC" (default), "DYNAMIC"

PHASEADJ: 0 (default), 22.5, 45, 67.5, 90, ..., 315, 337.5

DUTY: integers 2~14 (default: 8)

CLKOP_BYPASS: "DISABLED" (default), "ENABLED"

CLKOS_BYPASS: "DISABLED" (default), "ENABLED"

CLKOK_BYPASS: "DISABLED" (default), "ENABLED"

CLKOP_TRIM_POL: "FALLING" (default), "RISING"

CLKOP_TRIM_DELAY: integers 0~7 (default: 0)

CLKOS_TRIM_POL: "RISING" (default), "FALLING"

CLKOS_TRIM_DELAY: integers 0~3 (default: 0)

Description

EHXPLLE and EPLLD are the two primitives defined for XP2 GPLL. The EPLLD is used for both ECP2, and XP2 to support design migration. See EPLLD for details on migration. It is recommended to use EPLLD for all PLL configurations except for configurations involving Duty Trim Options, CLKOK2 and the CLKOS Fine Delay Port (WRDEL). Those features are only supported by the EHXPLLE primitive.

See the following table for port description.

Table 651:

Port	I/O	Function
CLKI	I	Input clock.
CLKFB	I	Feedback clock.
RST	I	PLL reset (connect to CNTRST port). High active reset.
RSTK	I	Reset for K divider (connect to RESETK port). High active reset.
CLKOP	O	PLL output clock (no phase shift).
CLKOS	O	PLL output clock (phase shifted/duty cycle changed).
CLKOK	O	PLL output to clock tree (no phase shift, low speed).
CLKOK2	O	PLL output clock (no phase shift, CLKOP/3).

Table 651:

Port	I/O	Function
LOCK	O	PLL LOCK to CLKI, asynchronous signal. Active high indicates PLL lock.
CLKINTFB	O	Internal feedback source. CLKOP divider output before CLOCK TREE.
WRDEL	I	Fine delay adjust (0 = no delay; 1 = ~70ps).
DRPAI[3:0]	I	Dynamic coarse phase shift, rising edge setting.
DFPAI[3:0]	I	Dynamic coarse phase shift, falling edge setting.
DDAMODE	I	DDA mode. Active high indicates pin control (DYNAMIC) and active low indicates fuse control (STATIC).

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

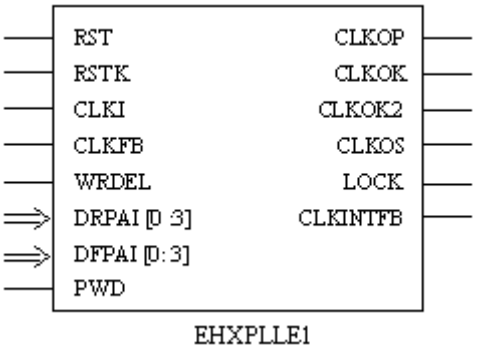
- TN1126 - LatticeXP2 sysCLOCK PLL Design and Usage Guide

EHXPLLE1

Complex PLL

Architectures Supported:

- LatticeXP2



INPUTS: CLKI, CLKFB, RST, RSTK, WRDEL, DRPAI3, DRPAI2, DRPAI1, DRPAI0, DFPAI3, DFPAI2, DFPAI1, DFPAI0, PWD

OUTPUTS: CLKOP, CLKOK, CLKOK2, CLKOS, LOCK, CLKINTFB

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKFB_DIV: integers 1~64 (default: 1)

CLKI_DIV: integers 1~64 (default: 1)

CLKOP_DIV: 2, 4, 8 (default), 16, 32, 48, 64, 80, 96, 112, 128

CLKOK_DIV: 2 (default), 4, 6, ..., 126, 128

PHASE_CNTL: "STATIC" (default), "DYNAMIC"

PHASEADJ: 0 (default), 22.5, 45, 67.5, 90, ..., 315, 337.5

DUTY: integers 2~14 (default: 8)

CLKOP_BYPASS: "DISABLED" (default), "ENABLED"

CLKOS_BYPASS: "DISABLED" (default), "ENABLED"

CLKOK_BYPASS: "DISABLED" (default), "ENABLED"

CLKOP_TRIM_POL: "FALLING" (default), "RISING"

CLKOP_TRIM_DELAY: integers 0~7 (default: 0)

CLKOS_TRIM_POL: "RISING" (default), "FALLING"

CLKOS_TRIM_DELAY: integers 0~3 (default: 0)

Description

The following are descriptions of EHXPLLE1 port functions.

Table 652:

Port	I/O	Function
CLKI	I	Input clock.
CLKFB	I	Feedback clock.
RST	I	PLL reset (connect to CNTRST port). High active reset.
RSTK	I	Reset for K divider (connect to RESETK port). High active reset.
WRDEL	I	Fine delay adjust (0 = no delay; 1 = ~70ps).
DRPAI[3:0]	I	Dynamic coarse phase shift, rising edge setting.
DFPAI[3:0]	I	Dynamic coarse phase shift, falling edge setting.
DDAMODE	I	DDA mode. Active high indicates pin control (DYNAMIC) and active low indicates fuse control (STATIC).
PWD	I	Dynamic power down signal.
CLKOP	O	PLL output clock (no phase shift).
CLKOS	O	PLL output clock (phase shifted/duty cycle changed).

Table 652:

Port	I/O	Function
CLKOK	O	PLL output to clock tree (no phase shift, low speed).
CLKOK2	O	PLL output clock (no phase shift, CLKOP/3).
LOCK	O	PLL LOCK to CLKI, asynchronous signal. Active high indicates PLL lock.
CLKINTFB	O	Internal feedback source. CLKOP divider output before CLOCK TREE.

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

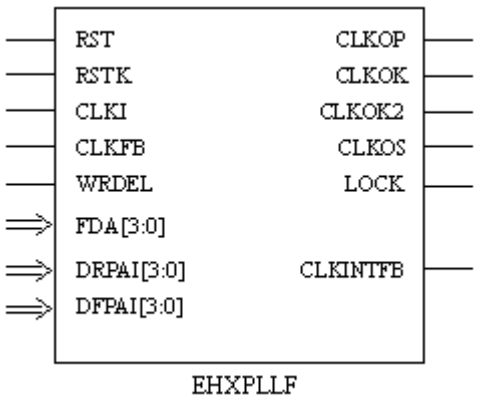
- TN1126 - LatticeXP2 sysCLOCK PLL Design and Usage Guide

EHXPLLF

Complex PLL

Architectures Supported:

- LatticeECP3



INPUTS: CLKI, CLKFB, RST, RSTK, DRPAI3, DRPAI2, DRPAI1, DRPAI0, DFPAI3, DFPAI2, DFPAI1, DFPAI0, FDA3, FDA2, FDA1, FDA0, WRDEL

OUTPUTS: CLKOP, CLKOS, CLKOK, CLKOK2, LOCK, CLKINTFB

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKI_DIV: integers 1~64 (default: 1)

CLKFB_DIV: integers 1~64 (default: 1)

CLKOP_DIV: 2, 4, 8 (default), 16, 32, 48, 64, 80, 96,112, 128

CLKOK_DIV: 2 (default), 4, 6, 8, ...,126,128

PHASEADJ: 0 (default), 22.5, 45, 67.5, 90, ..., 315, 337.5

DUTY: integers 2~14 (default: 8)

PHASE_DELAY_CNTL: "STATIC" (default), "DYNAMIC"

CLKOP_BYPASS: "DISABLED" (default), "ENABLED"

CLKOS_BYPASS: "DISABLED" (default), "ENABLED"

CLKOK_BYPASS: "DISABLED" (default), "ENABLED"

CLKOP_TRIM_POL: "RISING" (default), "FALLING"

CLKOP_TRIM_DELAY: integers 0~7 (default: 0)

CLKOS_TRIM_POL: "RISING" (default), "FALLING"

CLKOS_TRIM_DELAY: integers 0~3 (default: 0)

DELAY_VAL: integers 0~15 (default: 0)

DELAY_PWD: "DISABLED" (default), "ENABLED"

CLKOK_INPUT: "CLKOP" (default), "CLKOS"

Description

EHXPLLF and EPLLD are the two primitives defined for the LatticeECP3 GPLL. The EPLLD primitive is used for both ECP3, and XP2 for design migration. For the ECP3 new configurations, only EHXPLLF will be supported.

The following table describes EHXPLLF IO port functions.

Table 653:

Port	I/O	Function
CLKI	I	Input clock.
CLKFB	I	Feedback clock.
RST	I	PLL reset (connect to CNTRST port). High active reset.
RSTK	I	Reset for K divider (connect to RESETK port). High active reset.
CLKOP	O	PLL output clock (no phase shift).
CLKOS	O	PLL output clock (phase shifted/duty cycle changed).

Table 653:

Port	I/O	Function
CLKOK	O	PLL output clock (no phase shift, low speed).
CLKOK2	O	PLL output clock (no phase shift, CLKOP/3).
LOCK	O	PLL LOCK to CLKI, asynchronous signal. Active high indicates PLL lock.
CLKINTFB	O	Internal feedback source. CLKOP divider output before CLOCK TREE.
WRDEL	I	Dynamic CLKOS single step fine delay adjust (0 = no delay; 1 = ~70ps).
FDA[3:0]	I	Dynamic CLKOS 16 step fine delay adjustment on CLKOS (each increment is ~125ps).
DRPAI[3:0]	I	Dynamic coarse phase shift, rising edge setting.
DFPAI[3:0]	I	Dynamic duty cycle, falling edge setting.

You can refer to the following technical note on the Lattice web site for EHXPLLF port definition, attribute definition, and usage.

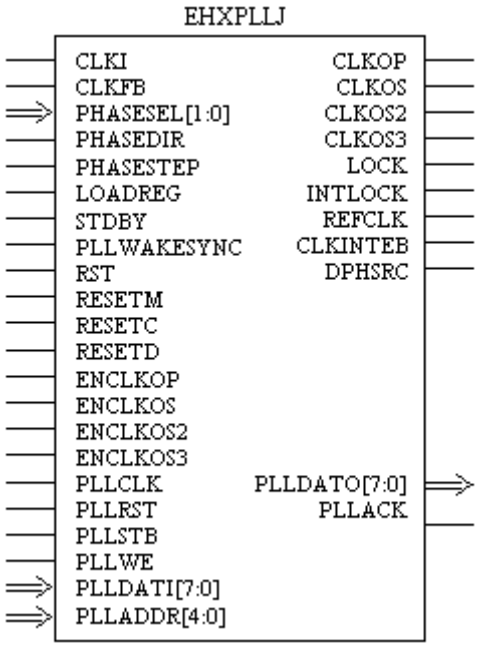
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide

EHXPLLJ

GPLL

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLKI, CLKFB, PHASESEL1, PHASESEL0, PHASEDIR, PHASESTEP, LOADREG, STDBY, PLLWAKESYNC, RST, RESETM, RESETC, RESETD, ENCLKOP, ENCLKOS, ENCLKOS2, ENCLKOS3, PLLCLK, PLLRST, PLLSTB, PLLWE, PLLDATI7, PLLDATI6, PLLDATI5, PLLDATI4, PLLDATI3, PLLDATI2, PLLDATI1, PLLDATI0, PLLADDR4, PLLADDR3, PLLADDR2, PLLADDR1, PLLADDR0

OUTPUTS: CLKOP, CLKOS, CLKOS2, CLKOS3, LOCK, INTLOCK, REFCLK, CLKINTFB, DPHSRC, PLLDATO7, PLLDATO6, PLLDATO5, PLLDATO4, PLLDATO3, PLLDATO2, PLLDATO1, PLLDATO0, PLLACK

ATTRIBUTES:

The EHXPLLJ primitive utilizes many attributes that allow the configuration of the PLL through source constraints. The following table details these attributes:

Table 654:

Attribute	Type	Allowed Values	Default	Description
FREQ_PIN_CLKI	String	10 to 400	100	CLKI frequency (MHz)
FREQ_PIN_CLKOP	String	3.125 to 400	100	CLKOP frequency (MHz)
CLKOP_FTOL	String	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	CLKOP frequency tolerance
CLKOP_AFREQ	String		-	CLKOP actual frequency (MHz)
FREQ_PIN_CLKOS	String	0.024 to 400	100	CLKOS frequency (MHz)

Table 654:

Attribute	Type	Allowed Values	Default	Description
CLKOS_FTOL	String	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	CLKOS frequency tolerance
CLKOS_AFREQ	String		-	CLKOS actual frequency (MHz)
FREQUENCY_PIN_CLKOS2	String	0.024 to 400	100	CLKOS2 frequency (MHz)
CLKOS2_FTOL	String	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	CLKOS2 frequency tolerance
CLKOS2_AFREQ	String		-	CLKOS2 actual frequency (MHz)
FREQUENCY_PIN_CLKOS3	String	0.024 to 400	100	CLKOS3 frequency (MHz)
CLKOS3_FTOL	String	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	CLKOS3 frequency tolerance
CLKOS3_AFREQ	String		-	CLKOS3 actual frequency (MHz)
CLKI_DIV	Integer	1 to 128	1	CLKI divider setting
CLKFB_DIV	Integer	1 to 128	1	CLKFB divider setting
CLKOP_DIV	Integer	1 to 128	8	CLKOP divider setting
CLKOS_DIV	Integer	1 to 128	8	CLKOS divider setting
CLKOS2_ DIV	Integer	1 to 128	8	CLKOS2 divider setting
CLKOS3_ DIV	Integer	1 to 128	8	CLKOS3 divider setting
CLOCK_ENABLE_PORTS	Boolean	ENABLED, DISABLED	DISABLED	Clock enable ports
CLKOP_ENABLE	Boolean	ENABLED, DISABLED	ENABLED	CLKOP enable
CLKOS_ENABLE	Boolean	ENABLED, DISABLED	ENABLED	CLKOS enable
CLKOS2_ENABLE	Boolean	ENABLED, DISABLED	ENABLED	CLKOS2 enable
CLKOS3_ENABLE	Boolean	ENABLED, DISABLED	ENABLED	CLKOS3 enable
VCO_BYPASS_A0	Boolean	ENABLE, DISABLED	DISABLED	VCO bypass A0
VCO_BYPASS_B0	Boolean	ENABLE, DISABLED	DISABLED	VCO bypass B0
VCO_BYPASS_C0	Boolean	ENABLE, DISABLED	DISABLED	VCO bypass C0
VCO_BYPASS_D0	Boolean	ENABLE, DISABLED	DISABLED	VCO bypass D0

Table 654:

Attribute	Type	Allowed Values	Default	Description
CLKOP _PHASEADJ	String	0, 45, 90, 135, 180, 225, 270, 315	0	CLKOP desired phase shift selection (O) in static mode
CLKOS_PHASEADJ	String	0, 45, 90, 135, 180, 225, 270, 315	0	CLKOS desired phase shift selection (O) in static mode
CLKOS2_PHASEADJ	String	0, 45, 90, 135, 180, 225, 270, 315	0	CLKOS2 desired phase shift selection (O) in static mode
CLKOS3_PHASEADJ	String	0, 45, 90, 135, 180, 225, 270, 315	0	CLKOS3 desired phase shift selection (O) in static mode
CLKOP_CPHASE	Integer	0 to 127	N/A	CLKOP coarse phase adjust
CLKOS_CPHASE	Integer	0 to 127	N/A	CLKOS coarse phase adjust
CLKOS2_CPHASE	Integer	0 to 127	N/A	CLKOS2 coarse phase adjust
CLKOS3_CPHASE	Integer	0 to 127	N/A	CLKOS3 coarse phase adjust
CLKOP_FPHASE	Integer	0 to 7	N/A	CLKOP fine phase adjust
CLKOS_FPHASE	Integer	0 to 7	N/A	CLKOS fine phase adjust
CLKOS2_FPHASE	Integer	0 to 7	N/A	CLKOS2 fine phase adjust
CLKOS3_FPHASE	Integer	0 to 7	N/A	CLKOS3 fine phase adjust
FEEDBK_PATH	String	CLKOP, CLKOS, CLKOS2, CLKOS3, INT_DIVA, INT_DIVB, INT_DIVC, INT_DIVD, USERCLOCK	CLKOP	Feedback mode
KVCO	Integer	0 to 7	0	VCO gain - Kvco
LPF_CAPACITOR	Integer	0 to 3	0	LPF capacitor
LPF_RESISTOR	Integer	0 to 127	0	LPF resistor
ICP_CURRENT	Integer	0 to 31	0	ICP current
FRACN_ENABLE	Boolean	ENABLE, DISABLED	DISABLED	Fractional-N divider enable
FRACN_DIV	Integer	0 to 65535	0	Fractional-N divider
FRACN_ORDER	Integer	0 to 3	0	Fractional-N noise shaping order
CLKOP_TRIM_POL	String	RISING, FALLING	RISING	CLKOP duty trim polarity
CLKOP_TRIM_DELAY	Integer	0, 1, 2, 4	0	CLKOP duty trim polarity delay
CLKOS_TRIM_POL	String	RISING, FALLING	RISING	CLKOS duty trim polarity
CLKOS_TRIM_DELAY	Integer	0, 1, 2, 4	0	CLKOS duty trim polarity delay

Table 654:

Attribute	Type	Allowed Values	Default	Description
PLL_EXPERT	Boolean	ENABLE, DISABLED	DISABLED	
PLL_USE_WB	Boolean	ENABLE, DISABLED	DISABLED	
PREDIVIDER_MUXA1	Integer	0 to 3	0	
PREDIVIDER_MUXB1	Integer	0 to 3	0	
PREDIVIDER_MUXC1	Integer	0 to 3	0	
PREDIVIDER_MUXD1	Integer	0 to 3	0	
OUTDIVIDER_MUXA2	String	DIVA, REFCLK	DIVA	
OUTDIVIDER_MUXB2	String	DIVB, REFCLK	DIVB	
OUTDIVIDER_MUXC2	String	DIVC, REFCLK	DIVC	
OUTDIVIDER_MUXD2	String	DIVD, REFCLK	DIVD	
FREQ_LOCK_ACCURACY	Integer	0 to 3	0	
PLL_LOCK_MODE	Integer	0 to 7	0	
PLL_LOCK_DELAY	Integer	1600, 800, 400, 200 (in ns)	200	
GMC_GAIN	Integer	0 to 7	0	GM/C gain
GMC_TEST	Integer	0 to 15	14	GM/C test mode
MFG1_TEST	Integer	0 to 7	0	
MFG2_TEST	Integer	0 to 7	0	
MFG_FORCE_VFILTER	Integer	0, 1	0	
MFG_ICP_TEST	Integer	0, 1	0	
MFG_ EN_UP	Integer	0, 1	0	
MFG_ FLOAT_ICP	Integer	0, 1	0	
MFG_GMC_PRESET	Integer	0, 1	0	
MFG_LF_PRESET	Integer	0, 1	0	
MFG_GMC_RESET	Integer	0, 1	0	
MFG_LF_RESET	Integer	0, 1	0	
MFG_LF_RESGRND	Integer	0, 1	0	
MFG_ GMCREF_SEL	Integer	0 to 3	2	
MFG_EN_FILTEROPAMP	Integer	0, 1	1	
STDBY_ENABLE	Boolean	ENABLED, DISABLED	DISABLED	

Table 654:

Attribute	Type	Allowed Values	Default	Description
REFIN_RESET	Boolean	ENABLED, DISABLED	DISABLED	
SYNC_ENABLE	Boolean	ENABLED, DISABLED	DISABLED	
INT_LOCK_STICKY	Boolean	ENABLED, DISABLED	DISABLED	
DPHASE_SOURCE	Boolean	ENABLED, DISABLED	DISABLED	
INTFB_WAKE	Boolean	ENABLED, DISABLED	DISABLED	
PLLRST_ENA	Boolean	ENABLED, DISABLED	DISABLED	
MRST_ENA	Boolean	ENABLED, DISABLED	DISABLED	
DCRST_ENA	Boolean	ENABLED, DISABLED	DISABLED	
DDRST_ENA	Boolean	ENABLED, DISABLED	DISABLED	

Description

EHXPLLJ is the GPLL primitive for MachXO2 and Platform Manager 2. A wrapper will be used around the primitive for configurations without the dynamic control or other ports. The EHXPLLJ primitive uses a single reference clock input.

For detailed information, refer to the following technical note on the Lattice web site.

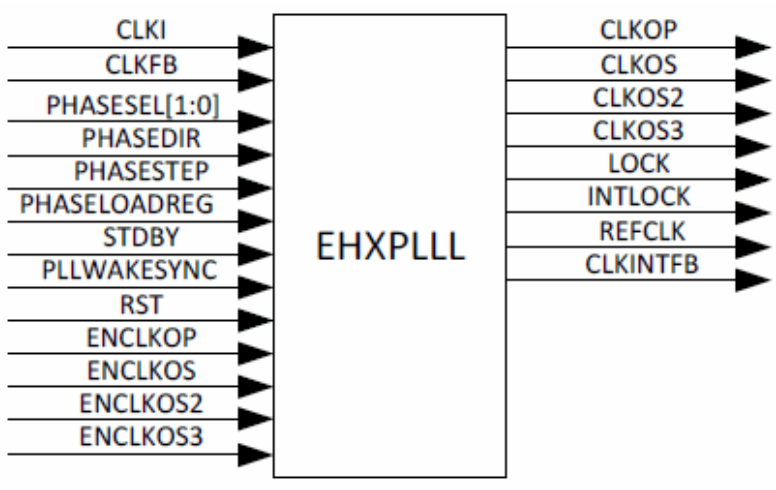
- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide

EHXPLLL

GPLL for ECP5

Architectures Supported:

- ECP5



INPUTS: CLKI, CLKFB, PHASESEL1, PHASESEL0, PHASEDIR, PHASESTEP, PHASELOADREG, STDBY, PLLWAKESYNC, RST, ENCLKOP, ENCLKOS, ENCLKOS2, ENCLKOS3

OUTPUTS: CLKOP, CLKOS, CLKOS2, CLKOS3, LOCK, INTLOCK, REFCLK, CLKINTFB

ATTRIBUTES:

The following table details EHXPLLL attributes.

Table 655:

Attribute Name	Type	Range	Default Value	Description
FREQUENCY_PIN_CLKI	NC		100	CLKI Frequency (MHz)
FREQUENCY_PIN_CLKOP				CLKOP Actual Frequency (MHz)
FREQUENCY_PIN_CLKOS				CLKOS Actual Frequency (MHz)
FREQUENCY_PIN_CLKOS2				CLKOS2 Actual Frequency (MHz)
FREQUENCY_PIN_CLKOS3				CLKOS3 Actual Frequency (MHz)
CLKI_DIV	C, S	1 – 128	1	CLKI Divider Setting
CLKFB_DIV	C, S	1 – 80	1	CLKFB Divider Setting
CLKOP_DIV	C, S	1 – 128	8	CLKOP Divider Setting
CLKOS_DIV	C, S	1 – 128	8	CLKOS Divider Setting
CLKOS2_ DIV	C, S	1 – 128	8	CLKOS2 Divider Setting
CLKOS3_ DIV	C, S	1 – 128	8	CLKOS3 Divider Setting
CLKOP_ENABLE	C, S	ENABLED, DISABLED	ENABLED	CLKOP Enable
CLKOS_ENABLE	C, S	ENABLED, DISABLED	DISABLED	CLKOS Enable

Table 655:

Attribute Name	Type	Range	Default Value	Description
CLKOS2_ENABLE	C, S	ENABLED, DISABLED	DISABLED	CLKOS2 Enable
CLKOS3_ENABLE	C, S	ENABLED, DISABLED	DISABLED	CLKOS3 Enable
CLKOP_CPHASE	C, S	0 – 127	N/A	CLKOP Coarse Phase adj
CLKOS_CPHASE	C, S	0 – 127	N/A	CLKOS Coarse Phase adj.
CLKOS2_CPHASE	C, S	0 – 127	N/A	CLKOS2 Coarse Phase adj.
CLKOS3_CPHASE	C, S	0 – 127	N/A	CLKOS3 Coarse Phase adj.
CLKOP_FPHASE	C, S	0 – 7	N/A	CLKOP Fine Phase adj.
CLKOS_FPHASE	C, S	0 – 7	N/A	CLKOS Fine Phase adj.
CLKOS2_FPHASE	C, S	0 – 7	N/A	CLKOS2 Fine Phase adj.
CLKOS3_FPHASE	C, S	0 – 7	N/A	CLKOS3 Fine Phase adj.
FEEDBK_PATH	C, S	CLKOP, CLKOS, CLKOS2, CLKOS3, INT_OP, INT_OS, INT_OS2, INT_OS3, USERCLOCK	CLKOP	Feedback Mode
KVCO	C	0 – 7	0	VCO Gain - Kvco
LPF_CAPACITOR	C	0 – 3	0	LPF Capacitor
LPF_RESISTOR	C	0 – 127	0	LPF Resistor
ICP_CURRENT	C	0 – 31	0	ICP Current
CLKOP_TRIM_POL	C	RISING, FALLING	RISING	CLKOP Duty Trim Polarity
CLKOP_TRIM_DELAY	C	0, 1, 2, 4	0	CLKOP Duty Trim Polarity Multiplier
CLKOS_TRIM_POL	C	RISING, FALLING	RISING	CLKOS Duty Trim Polarity
CLKOS_TRIM_DELAY	C	0, 1, 2, 4	0	CLKOS Duty Trim Polarity Multiplier
OUTDIVIDER_MUXA	C, S	DIVA, REFCLK	DIVA	OUTDIVIDER_MUXA
OUTDIVIDER_MUXB	C, S	DIVB, REFCLK	DIVB	OUTDIVIDER_MUXB
OUTDIVIDER_MUXC	C, S	DIVC, REFCLK	DIVC	OUTDIVIDER_MUXC
OUTDIVIDER_MUXD	C, S	DIVD, REFCLK	DIVD	OUTDIVIDER_MUXD
FREQ_LOCK_ACCURACY	C	0 – 3	0	FREQ_LOCK_ACCURACY
PLL_LOCK_MODE	C, S	0 – 7	0	PLL_LOCK_MODE
PLL_LOCK_DELAY	S	1600, 800, 400, 200 ns	200	PLL_LOCK_DELAY
MFG_GMC_GAIN	C	0 – 7	0	GM/C Gain
MFG_GMC_TEST	C	0 – 15	14	GM/C Test Mode
MFG1_TEST	C	0 – 7	0	MFG1_TEST
MFG2_TEST	C	0 – 7	0	MFG2_TEST

Table 655:

Attribute Name	Type	Range	Default Value	Description
MFG_FORCE_VFILTER	C	0 – 1	0	MFG_FORCE_VFILTER
MFG_ICP_TEST	C	0 – 1	0	MFG_ICP_TEST
MFG_ EN_UP	C	0 – 1	0	MFG_EN_UP
MFG_ FLOAT_ICP	C	0 – 1	0	MFG_FLOAT_ICP
MFG_GMC_PRESET	C	0 – 1	0	MFG_GMC_PRESET
MFG_LF_PRESET	C	0 – 1	0	MFG_LF_PRESET
MFG_GMC_RESET	C	0 – 1	0	MFG_GMC_RESET
MFG_LF_RESET	C	0 – 1	0	MFG_LF_RESET
MFG_LF_RESGRND	C	0 – 1	0	MFG_LF_RESGRND
MFG_ GMCREF_SEL	C	0 – 3	2	MFG_ GMCREF_SEL
MFG_EN_FILTEROPAMP	C	0 – 1	1	MFG_EN_FILTEROPAMP
STDBY_ENABLE	C	ENABLED, DISABLED	DISABLED	STDBY_ENABLE
REFIN_RESET	S	ENABLED, DISABLED	DISABLED	REFIN_RESET
SYNC_ENABLE	S	ENABLED, DISABLED	DISABLED	SYNC_ENABLE
INT_LOCK_STICKY	S	ENABLED, DISABLED	ENABLED	INT_LOCK_STICKY
DPHASE_SOURCE	C, S	ENABLED, DISABLED	DISABLED	DPHASE_SOURCE
PLLRST_ENA	C, S	ENABLED, DISABLED	DISABLED	Enable PLL Reset
INTFB_WAKE	C	ENABLED, DISABLED	DISABLED	Internal Feedback on Wakeup
MFG_VCO_NORESET	C	0 – 1	0	Disable VCO reset in STDBY mode
MFG_STDBY_ANALOGON	C	0 – 1	0	Enable analog bias in STDBY mode
MFG_NO_PLLRESET	C	0 – 1	0	Disable pll_rst_n in STDBY mode

Description

EHXPLLL is the GPLL primitive for ECP5. The following are descriptions of EHXPLLL port functions.

Table 656:

Port	I/O	Function
CLKI	I	Input Clock to PLL.
CLKI2	I	Muxed Input Clock to PLL.
SEL	I	Select Clock
CLKFB	I	Feedback Clock.
PHASESEL[1:0]	I	Select the output affected by Dynamic Phase adjustment.
PHASEDIR	I	Dynamic Phase adjustment direction.

Table 656:

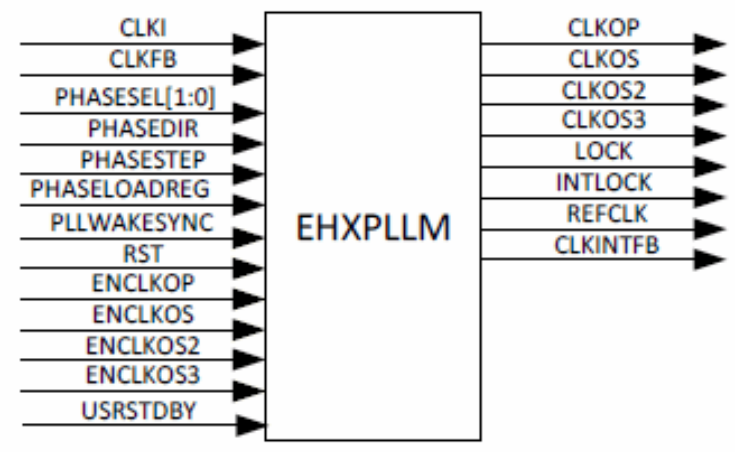
Port	I/O	Function
PHASESTEP	I	Dynamic Phase adjustment step.
PHASELOADREG	I	Load dynamic phase adjustment values into PLL.
CLKOP	O	PLL main output clock.
CLKOS	O	PLL output clock.
CLKOS2	O	PLL output clock.
CLKOS3	O	PLL output clock.
LOCK	O	PLL LOCK to CLKI, Asynchronous signal. Active high indicates PLL lock.
STDBY	I	Standby signal to power down the PLL.
RST	I	Resets the whole PLL.
ENCLKOP	I	Enable PLL output CLKOP
ENCLKOS	I	Enable PLL output CLKOS
ENCLKOS2	I	Enable PLL output CLKOS2
ENCLKOS3	I	Enable PLL output CLKOS3

EHXPLLM

GPLL

Architectures Supported:

- LIFMD
- LIFMDF



INPUTS: CLKI, CLKFB, PHASESEL1, PHASESEL0, PHASEDIR, PHASESTEP, PHASELOADREG, STDBY, PLLWAKESYNC, RST, ENCLKOP, ENCLKOS, ENCLKOS2, ENCLKOS3, USRSTDBY

OUTPUTS: CLKOP, CLKOS, CLKOS2, CLKOS3, LOCK, INTLOCK, REFCLK, CLKINTFB

EHXPLLM is the GPLL primitive for LIFMD/F. The following are descriptions of EHXPLLM port functions

Table 657:

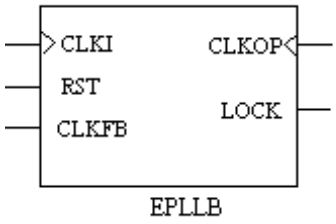
Port	IO	Description
CLKI	I	Input Clock to PLL.
CLKI2	I	Muxed Input Clock to PLL. Note 1.
SEL	I	Select Clock. Note 1.
CLKFB	I	Feedback Clock.
PHASESEL[1:0]	I	Select the output affected by Dynamic Phase adjustment.
PHASEDIR	I	Dynamic Phase adjustment direction.
PHASESTEP	I	Dynamic Phase adjustment step.
PHASELOADREG	I	Load dynamic phase adjustment values into PLL.
CLKOP	O	PLL main output clock.
CLKOS	O	PLL output clock.
CLKOS2	O	PLL output clock.
CLKOS3	O	PLL output clock.
LOCK	O	PLL LOCK to CLKI, Asynchronous signal. Active high indicates PLL lock.
RST	I	Resets the whole PLL.
ENCLKOP	I	Enable PLL output CLKOP
ENCLKOS	I	Enable PLL output CLKOS
ENCLKOS2	I	Enable PLL output CLKOS2
ENCLKOS3	I	Enable PLL output CLKOS3
USRSTDBY	I	User port to put the PLL in sleep mode

EPLLB

Enhanced PLL

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: CLKI, RST, CLKFB

OUTPUTS: CLKOP, LOCK

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKI_DIV: (*LatticeECP/EC*) integers 1~16 (default: 1);
(*LatticeXP*) integers 1~15 (default: 1)

CLKFB_DIV: (*LatticeECP/EC*) integers 1~16 (default: 1);
(*LatticeXP*) integers 1~15 (default: 1)

CLKOP_DIV: (*LatticeECP/EC*) even integers 2~32 if CLKOS is not used; 2, 4, 8, 16, 32 is CLKOS is used. (Default: 8)
(*LatticeXP*) even integers 2~30 if CLKOS is not used (default: 6); 2, 4, 8, 16 if CLKOS is used (default: 4).

FDEL: integers -8~8 (default: 0)

WAKE_ON_LOCK: "OFF" (default), "ON"

FB_MODE: "CLOCKTREE" (default), "INTERNAL", "EXTERNAL"

LOCK_CYC: integer (default: 2)

Description

The following are descriptions of EPLLB port functions.

Table 658:

Port	I/O	Function
CLKI	I	Global clock input; frequency: 20~420 MHz.
RST	I	PLL reset.
CLKFB	I	External feedback, internal feedback from CLKOP divider; frequency: 20~420 MHz.

Table 658:

Port	I/O	Function
CLKOP	O	PLL output clock to clock tree (no phase shift); frequency: 20~420 MHz.
LOCK	O	"1" indicates PLL LOCK to CLK_IN.

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

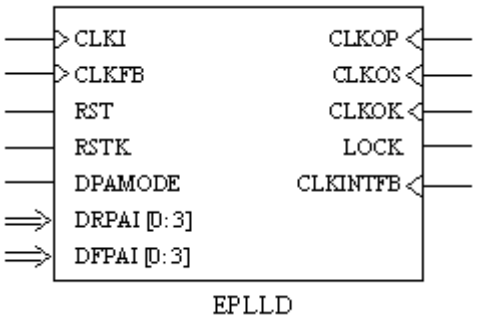
- TN1049 - LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide

EPLLD

Enhanced PLL

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: CLKI, CLKFB, RST, RSTK, DPAMODE, DRPAI3, DRPAI2, DRPAI1, DRPAI0, DFPAI3, DFPAI2, DFPAI1, DFPAI0

OUTPUTS: CLKOP, CLKOS, CLKOK, LOCK, CLKINTFB

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKI_DIV: integers 1~64 (default: 1)

CLKFB_DIV: integers 1~64 (default: 1)

CLKOP_DIV: 2, 4, 8 (default), 16, 32, 48, 64, 80, 96,112, 128

CLKOK_DIV: 2 (default), 4, 6, 8, ..., 126, 128

PHASEADJ: 0 (default), 22.5, 45, 67.5, 90, ..., 315, 337.5

DUTY: integers 2~14 (default: 8)

PHASE_CNTL: "STATIC" (default), "DYNAMIC"

PLLCAP: "DISABLED" (default), "ENABLED", "AUTO"

CLKOP_BYPASS: "DISABLED" (default), "ENABLED"

CLKOS_BYPASS: "DISABLED" (default), "ENABLED"

CLKOK_BYPASS: "DISABLED" (default), "ENABLED"

PLLTYPE: "AUTO" (default), "SPLL", "GPLL"

Description

The following are descriptions of EPLLD port functions.

Table 659:

Port	I/O	Function
CLKI	I	Input clock.
CLKFB	I	Feedback clock.
RST	I	PLL reset (connected to the CNTRST port). High active reset.
RSTK	I	Reset for K divider (connected to the RESETK port). High active reset.
DPAMODE	I	Dynamic phase adjust mode. Active high indicates pin control (DYNAMIC) and active low indicates fuse control (STATIC).
DRPAI[3:0]	I	Dynamic coarse phase shift; rise edge setting.
DFPAI[3:0]	I	Dynamic coarse phase shift; falling edge setting.
CLKOP	O	PLL output clock (no phase shift).
CLKOS	O	PLL output clock (phase shifted/duty cycle changed).
CLKOK	O	PLL output to clock tree (no phase shift, low speed).
LOCK	O	PLL LOCK to CLKI, asynchronous signal. Active high indicates PLL lock.
CLKINTFB	O	Internal feedback source. CLKOP divider output before CLOCK TREE.

The EPLLD primitive can be used for design migration between LatticeECP2. and LatticeXP2, which can be divided into four situations:

Design Migration From LatticeECP2 to LatticeXP2 (EPLLD Configurations):

- EPLLD Configurations without Dynamic Phase & Duty Options: The LatticeECP2 PLL configuration migrates to LatticeXP2 without any changes.
- EPLLD Configurations with Dynamic Phase & Duty Options: The LatticeECP2 PLL configuration has DPAMODE port in the top-level port list. To migrate this configuration to LatticeXP2, the user has to tie the DPAMODE port to GND.

Design Migration From LatticeECP2 to LatticeXP2 (EHXPLLD Configurations): This configuration cannot be migrated to LatticeXP2 because LatticeXP2 does not support Delay Adjust.

Design Migration From LatticeXP2 to LatticeECP2 (EPLLD Configurations):

- EPLLD Configurations without Dynamic Phase & Duty Options: The LatticeXP2 PLL configuration migrates to LatticeECP2 without any changes.
- EPLLD Configurations with Dynamic Phase & Duty Options (No Duty Trim): The LatticeXP2 PLL configuration has no DPAMODE port in the top-level port list. Two options for migration:
  a. Regenerate the PLL configuration for LatticeECP2.
  b. Modify the LatticeXP2 PLL configuration to bring the DPAMODE to top-level port list.

Design Migration From LatticeXP2 to LatticeECP2 (EHXPLLE Configurations): This configuration cannot be migrated to LatticeECP2 because LatticeECP2 does not support Duty Trim Options, CLKOK2 and the CLKOS Fine Delay Port (WRDEL).

You can refer to the following technical notes on the Lattice web site for more detailed description and usage.

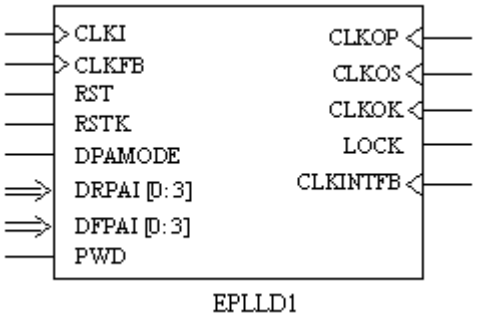
- TN1103 - LatticeECP2 sysCLOCK PLL/DLL Design and Usage Guide
- TN1126 - LatticeXP2 sysCLOCK PLL Design and Usage Guide

EPLLD1

Enhanced PLL

Architectures Supported:

- LatticeXP2



INPUTS: CLKI, CLKFB, RST, RSTK, DPAMODE, DRPAI3, DRPAI2, DRPAI1, DRPAI0, DFPAI3, DFPAI2, DFPAI1, DFPAI0, PWD

OUTPUTS: CLKOP, CLKOS, CLKOK, LOCK, CLKINTFB

ATTRIBUTES:

FIN: 20.0000~420.0000 (in MHz) (default: "100.0000")

CLKI_DIV: integers 1~64 (default: 1)

CLKFB_DIV: integers 1~64 (default: 1)

CLKOP_DIV: 2, 4, 8 (default), 16, 32, 48, 64, 80, 96,112, 128

CLKOK_DIV: 2 (default), 4, 6, 8, ..., 126, 128

PHASEADJ: 0 (default), 22.5, 45, 67.5, 90, ..., 315, 337.5

DUTY: integers 2~14 (default: 8)

PHASE_CNTL: "STATIC" (default), "DYNAMIC"

PLLCAP: "DISABLED" (default), "ENABLED", "AUTO"

CLKOP_BYPASS: "DISABLED" (default), "ENABLED"

CLKOS_BYPASS: "DISABLED" (default), "ENABLED"

CLKOK_BYPASS: "DISABLED" (default), "ENABLED"

PLLTYPE: "AUTO" (default), "SPLL", "GPLL"

Description

The following are descriptions of EPLLD1 port functions.

Table 660:

Port	I/O	Function
CLKI	I	Input clock.
CLKFB	I	Feedback clock.
RST	I	PLL reset (connected to the CNTRST port). High active reset.
RSTK	I	Reset for K divider (connected to the RESETK port). High active reset.
DPAMODE	I	Dynamic phase adjust mode. Active high indicates pin control (DYNAMIC) and active low indicates fuse control (STATIC).
DRPAI[3:0]	I	Dynamic coarse phase shift; rise edge setting.
DFPAI[3:0]	I	Dynamic coarse phase shift; falling edge setting.
PWD	I	Dynamic power down signal.
CLKOP	O	PLL output clock (no phase shift).
CLKOS	O	PLL output clock (phase shifted/duty cycle changed).
CLKOK	O	PLL output to clock tree (no phase shift, low speed).
LOCK	O	PLL LOCK to CLKI, asynchronous signal. Active high indicates PLL lock.
CLKINTFB	O	Internal feedback source. CLKOP divider output before CLOCK TREE.

You can refer to the following technical note on the Lattice web site for more detailed description and usage.

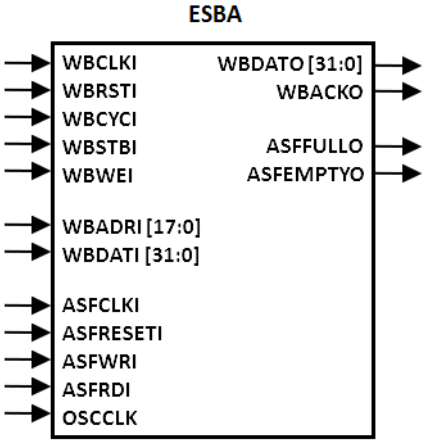
- TN1126 - LatticeXP2 sysCLOCK PLL Design and Usage Guide

ESBA

Embedded Security Block including a WishBone Interface.

Architectures Supported:

- LFMNX
- MachXO3D



INPUTS: WBDATI31, WBDATI30, WBDATI29, WBDATI28, WBDATI27, WBDATI26, WBDATI25, WBDATI24, WBDATI23, WBDATI22, WBDATI21, WBDATI20, WBDATI19, WBDATI18, WBDATI17, WBDATI16, WBDATI15, WBDATI14, WBDATI13, WBDATI12, WBDATI11, WBDATI10, WBDATI9, WBDATI8, WBDATI7, WBDATI6, WBDATI5, WBDATI4, WBDATI3, WBDATI2, WBDATI1, WBDATI0, WBSTBI, WBCLKI, ASFCLKI, WBRSTI, ASFRESETI, WBADRI17, WBADRI16, WBADRI15, WBADRI14, WBADRI13, WBADRI12, WBADRI11, WBADRI10, WBADRI9, WBADRI8, WBADRI7, WBADRI6, WBADRI5, WBADRI4, WBADRI3, WBADRI2, WBADRI1, WBADRI0, WBCYCI, WBWEI, ASFWRI, ASFRDI, OSCCLK

OUTPUTS: WBDATO31, WBDATO30, WBDATO29, WBDATO28, WBDATO27, WBDATO26, WBDATO25, WBDATO24, WBDATO23, WBDATO22, WBDATO21, WBDATO20, WBDATO19, WBDATO18, WBDATO17, WBDATO16, WBDATO15, WBDATO14, WBDATO13, WBDATO12, WBDATO11, WBDATO10, WBDATO9, WBDATO8, WBDATO7, WBDATO6, WBDATO5, WBDATO4, WBDATO3, WBDATO2, WBDATO0, WBACKO, ASFFULLO, ASFEMPTYO

ATTRIBUTES:

UDS_TRN: 0 (default)

UDS_TRN_FORMAT: ASCII (default)

Table 661:

Port	I/O	Description
WBDATI[31:0]	I	Write Data Input. Split transactions are not supported requiring the master to stay active on the bus until a valid response is received.
WBDATO[31:0]	O	Read Data Output
WBSTBI	I	Strobe input indicating the WB slave is the target for the current transaction on the WB bus.

Table 661:

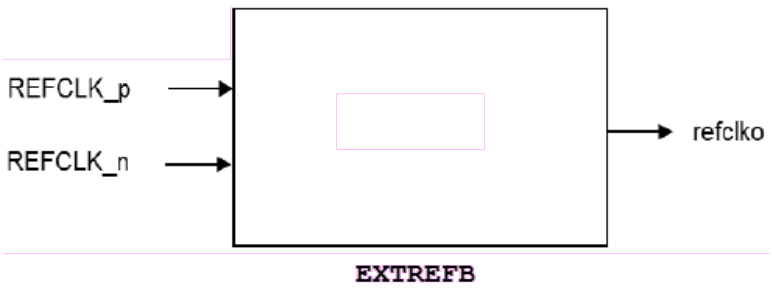
Port	I/O	Description
WBACKO	O	Transfer Acknowledge asserted by the WB slave to the master, indicating the requested transfer has been completed. This signal is qualified by WBSTBI.
WBCLKI	I	Positive edge clock used by all WB interface logic blocks.
ASFFULLO	O	Indicates FIFO is full.
ASFCLKI	I	Clock for ASF.
ASFEMPTYO	O	Indicates FIFO is empty.
WBRSTI	I	Resets the WB logic.
ASFRESETI	I	Resets the ASF logic.
WBADRI[17:0]	I	Address
WBCYCI	I	Cycle input indicates the WB slave a valid bus cycle is present on the bus. In multiple-master configuration, this signal serves as a bus request.
WBWEI	I	Write/Read indicator. 0 : Read transaction; 1 : Write transaction
ASFWRI	I	Write strobe
ASFRDI	I	Read strobe
OSCCLK	I	Input clock used for ESB

EXTREFB

Reference clock input buffer primitive for the dedicated external clock inputs to the Serdes TxPLL

Architectures Supported:

- ECP5



INPUTS: REFCLK_p, REFCLK_n

OUTPUTS: REFCLKO

The table below describes the I/O ports of the EXTREFA primitive.

Table 662:

Port	I/O	Function
REFCLKP	I	External reference clock positive input port.
REFCLKN	I	External reference clock negative input port
REFCLKO	O	Single-ended clock signal to be connected to PCS PLL inputs.

Description

The EXTREFB module takes the differential external reference clock pins (refclkp/n) and sends out a single-ended clock signal to the SerDes TxPLL. The EXTREFB can only connect to SerDes TxPLL in the same DCUA, or to SerDes TxPLL in the complementary sharing DCUA.

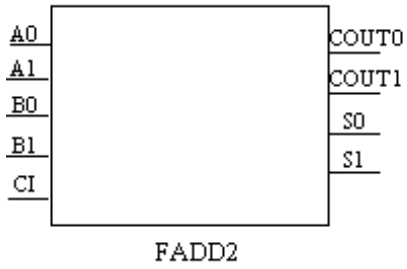
F

FADD2

2 Bit Fast Adder

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- MachXO
- Platform Manager



INPUTS: A1, A0, B1, B0, CI

OUTPUTS: COUT1, COUT0, S1, S0

Description

FADD2 is a 2-bit adder. It has a carry-in input (CI) and two 2-bit input (A0, A1 and B0, B1). The FADD2 produces a 2-bit sum output (S0, S1) along with a 2-bit carry-out output (COUT1, COUT).

Example pin functions:

Table 663:

Function	Pins
input	A1, A0, B1, B0
output	S1, S0
carry-in input	CI

Table 663:

Function	Pins
carry-out output (Bit-0)	COUT0
carry-out output (Bit-1)	COUT1

Truth Table

Table 664:

INPUTS					OUTPUTS			
A0	A1	B0	B1	CI	S0	COUT0	S1	COUT1
0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1	0
0	0	1	1	0	1	0	1	0
1	1	1	1	0	0	1	1	1
0	0	0	0	1	1	0	0	0
1	1	0	0	1	0	1	0	1
0	0	1	1	1	0	1	0	1
1	1	1	1	1	1	1	1	1

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

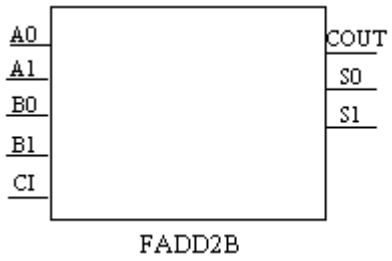
FADD2B

Fast 2 Bit Adder

Architectures Supported:

- LatticeECP2/M
- LatticeECP3
- LatticeXP2
- LFMNX
- MachXO2
- MachXO3D
- MachXO3L

- Platform Manager 2



INPUTS: A0, A1, B0, B1, CI

OUTPUTS: COUT, S0, S1

Note

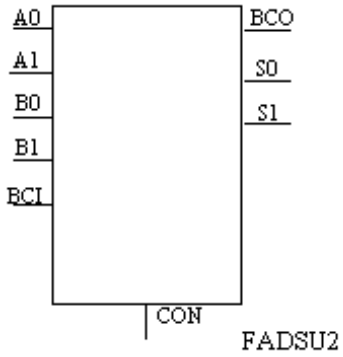
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FADSU2

2 Bit Fast Adder/Subtractor (two's complement)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A0, A1, B0, B1, BCI, CON

OUTPUTS: BCO, S0, S1

Description

FADSU2 is a 2 bit adder/subtractor. When the control signal (CON) is high FADSU2 functions as a 2 bit adder with a carry-in input (BCI) and two 2 bit inputs (A0:A1 and B0:B1), producing a 2 bit SUM output (S0:S1) along with a carry-out output (BCO).

When the control signal (CON) is low, FADSU2 functions as a 2 bit two's complement subtractor with a borrow-in input (BCI) and two 2 bit inputs (A0:A1 and B0:B1), producing a 2 bit two's complement output of A minus B (S0:S1) along with a borrow-out output (BCO).

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

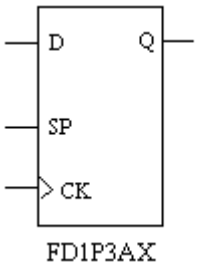
FD1P3AX

Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR Used for Clear

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2

- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, SP, CK

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 665:

INPUTS			OUTPUTS
D	SP	CK	Q
X	0	X	Q
0	1	↑	0
1	1	↑	1

X = Don’t care

When GSR=0, Q=0 (D=SP=CK=X)

Note

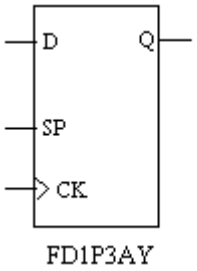
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1P3AY

Positive Edge Triggered D Flip-Flop with Positive Level Enable, GSR Used for Preset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, SP, CK

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 666:

INPUTS			OUTPUTS
D	SP	CK	Q
X	0	X	Q
0	1	↑	0
1	1	↑	1

X = Don’t care

When GSR=0, Q=1 (D=SP=CK=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

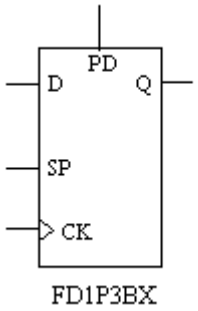
FD1P3BX

Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D

- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, SP, CK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 667:

INPUTS				OUTPUTS
D	SP	CK	PD	Q
X	0	X	0	Q
X	X	X	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=1 (D=SP=CK=PD=X)

Note

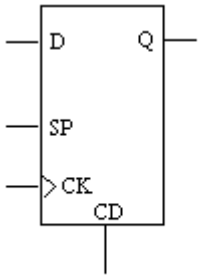
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1P3DX

Positive Edge Triggered D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



FD1P3DX

INPUTS: D, SP, CK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 668:

INPUTS				OUTPUTS
D	SP	CK	CD	Q
X	0	X	0	Q
X	X	X	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=SP=CK=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

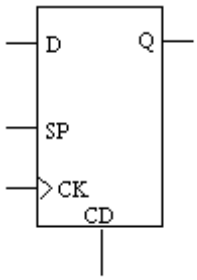
FD1P3IX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L

- Platform Manager
- Platform Manager 2



FD1P3IX

INPUTS: D, SP, CK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 669:

INPUTS				OUTPUTS
D	SP	CK	CD	Q
X	0	X	0	Q
X	X	↑	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=SP=CK=CD=X)

Note

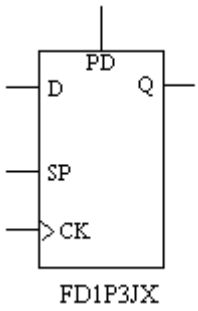
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1P3JX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, SP, CK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 670:

INPUTS				OUTPUTS
D	SP	CK	PD	Q
X	0	X	0	Q
X	X	↑	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=1 (D=SP=CK=PD=X)

Note

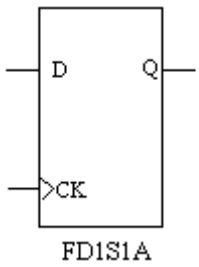
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1S1A

Positive Level Data Latch with GSR Used for Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, CK

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 671:

INPUTS		OUTPUTS
D	CK	Q
X	0	Q
0	1	0
1	1	1

X = Don’t care

When GSR=0, Q=0 (D=CK=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

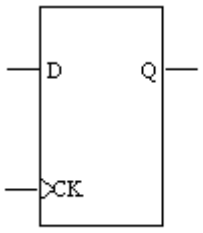
FD1S1AY

Positive Level Data Latch with GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP

- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



FD1S1AY

INPUTS: D, CK

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 672:

INPUTS		OUTPUTS
D	CK	Q
X	0	Q
0	1	0
1	1	1

X = Don’t care

When GSR=0, Q=1 (D=CK=X)

Note

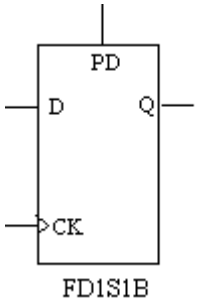
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1S1B

Positive Level Data Latch with Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, CK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 673:

INPUTS			OUTPUTS
D	CK	PD	Q
X	0	0	Q
X	X	1	1
0	1	0	0
1	1	0	1

X= Don’t care

When GSR=0, Q=1 (D=CK=PD=X)

Note

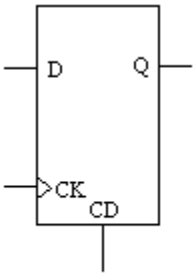
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1S1D

Positive Level Data Latch with Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



FD1S1D

INPUTS: D, CK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 674:

INPUTS			OUTPUTS
D	CK	CD	Q
X	0	0	Q
X	X	1	0
0	1	0	0
1	1	0	1

X = Don’t care

When GSR=0, Q=0 (D=CK=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

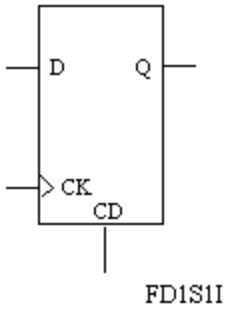
FD1S1I

Positive Level Data Latch with Positive Level Synchronous Clear

Architectures Supported:

- LatticeECP/EC

- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



FD1S1I

INPUTS: D, CK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 675:

INPUTS			OUTPUTS
D	CK	CD	Q
X	0	0	Q
X	1	1	0
0	1	0	0
1	1	0	1

X = Don’t care

When GSR=0, Q=0 (D=CK=CD=X)

Note

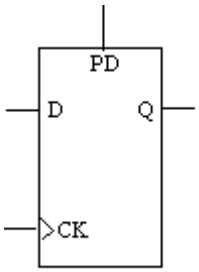
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1S1J

Positive Level Data Latch with Positive Level Synchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



FD1S1J

INPUTS: D, CK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 676:

INPUTS			OUTPUTS
D	CK	PD	Q
X	0	0	Q
X	1	1	1
0	1	0	0
1	1	0	1

X = Don’t care

When GSR=0, Q=1 (D=CK=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

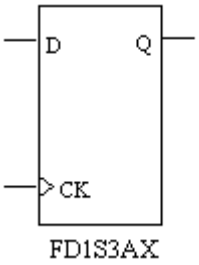
FD1S3AX

Positive Edge Triggered D Flip-Flop, GSR Used for Clear

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager

- Platform Manager 2



INPUTS: D, CK

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 677:

INPUTS		OUTPUTS
D	CK	Q
0	↑	0
1	↑	1

X = Don’t care

When GSR=0, Q=0 (D=CK=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

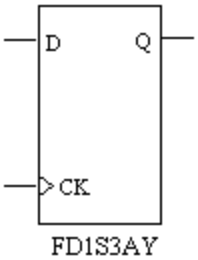
FD1S3AY

Positive Edge Triggered D Flip-Flop, GSR Used for Preset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M

- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, CK

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 678:

INPUTS		OUTPUTS
D	CK	Q
0	↑	0
1	↑	1

X = Don’t care

When GSR=0, Q=1 (D=CK=X)

Note

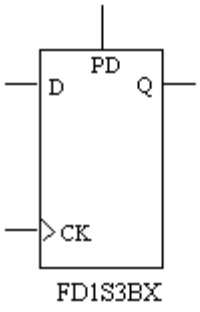
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1S3BX

Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Preset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, CK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 679:

INPUTS			OUTPUTS
D	CK	PD	Q
X	X	1	1
0	↑	0	0
1	↑	X	1

X = Don’t care

When GSR=0, Q=1 (D=CK=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

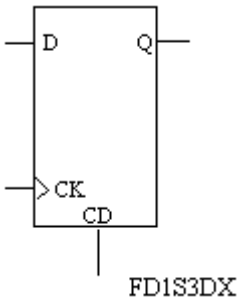
FD1S3DX

Positive Edge Triggered D Flip-Flop with Positive Level Asynchronous Clear

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, CK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 680:

INPUTS			OUTPUTS
D	CK	CD	Q
X	X	1	0
0	↑	0	0
1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=CK=CD=X)

Note

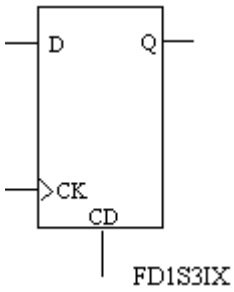
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FD1S3IX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D, CK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 681:

INPUTS			OUTPUTS
D	CK	CD	Q
X	↑	1	0
0	↑	0	0
1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=CK=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

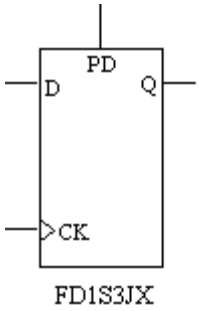
FD1S3JX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager

- Platform Manager 2



INPUTS: D, CK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 682:

INPUTS			OUTPUTS
D	CK	PD	Q
X	↑	1	1
0	↑	0	0
1	↑	X	1

X = Don’t care

When GSR=0, Q=1 (D=CK=PD=X)

Note

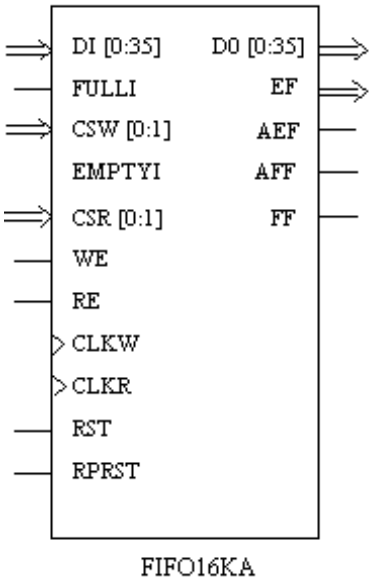
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FIFO16KA

16K FIFO

Architectures Supported:

- LatticeSC/M



FIFO16KA

INPUTS: DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, DI18, DI19, DI20, DI21, DI22, DI23, DI24, DI25, DI26, DI27, DI28, DI29, DI30, DI31, DI32, DI33, DI34, DI35, FULLI, CSW0, CSW1, EMPTYI, CSR0, CSR1, WE, RE, CLKW, CLKR, RST, RPRST

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17, DO18, DO19, DO20, DO21, DO22, DO23, DO24, DO25, DO26, DO27, DO28, DO29, DO30, DO31, DO32, DO33, DO34, DO35, EF, AEF, AFF, FF

ATTRIBUTES:

DATA_WIDTH_W: 1, 2, 4, 9, 18 (default), 36

DATA_WIDTH_R: 1, 2, 4, 9, 18 (default), 36

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 2-bit binary value (default: 2'b00)

CSDECODE_R: any 2-bit binary value (default: 2'b00)

GSR: "DISABLED" (default), "ENABLED"

AEPOINTER: any 15-bit binary value (default: all zeros)

AEPOINTER1: any 15-bit binary value (default: all zeros)

AFPOINTER: any 15-bit binary value (default: all zeros)

AFPOINTER1: any 15-bit binary value (default: all zeros)

FULLPOINTER: any 15-bit binary value (default: all zeros)

FULLPOINTER1: any 15-bit binary value (default: all zeros)

Description

You can refer to the following technical note on the Lattice web site on details of EBR port definition, attribute definition and usage.

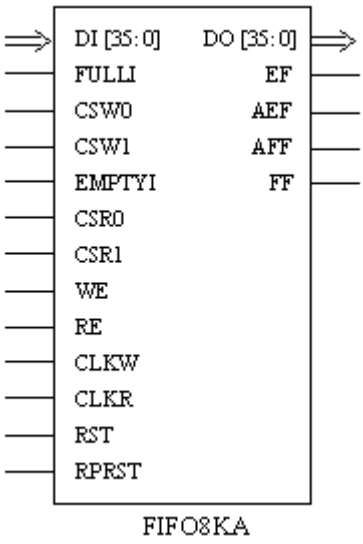
- TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices

FIFO8KA

8K FIFO

Architectures Supported:

- MachXO
- Platform Manager



INPUTS: DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, DI18, DI19, DI20, DI21, DI22, DI23, DI24, DI25, DI26, DI27, DI28, DI29, DI30, DI31, DI32, DI33, DI34, DI35, FULLI, CSW0, CSW1, EMPTYI, CSR0, CSR1, WE, RE, CLKW, CLKR, RST, RPRST

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17, DO18, DO19, DO20, DO21,

DO22, DO23, DO24, DO25, DO26, DO27, DO28, DO29, DO30, DO31, DO32, DO33, DO34, DO35, EF, AEF, AFF, FF

ATTRIBUTES:

DATA_WIDTH_W: 1, 2, 4, 9, 18 (default), 36

DATA_WIDTH_R: 1, 2, 4, 9, 18 (default), 36

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 2-bit binary value (default: 2'b00)

CSDECODE_R: any 2-bit binary value (default: 2'b00)

AEPOINTER: any 14-bit binary value (default: all zeros)

AEPOINTER1: any 14-bit binary value (default: all zeros)

AFPOINTER: any 14-bit binary value (default: all zeros)

AFPOINTER1: any 14-bit binary value (default: all zeros)

FULLPOINTER: any 14-bit binary value (default: all zeros)

FULLPOINTER1: any 14-bit binary value (default: all zeros)

GSR: "DISABLED" (default), "ENABLED"

Description

You can refer to the following technical note on the Lattice web site on detailed information and usage.

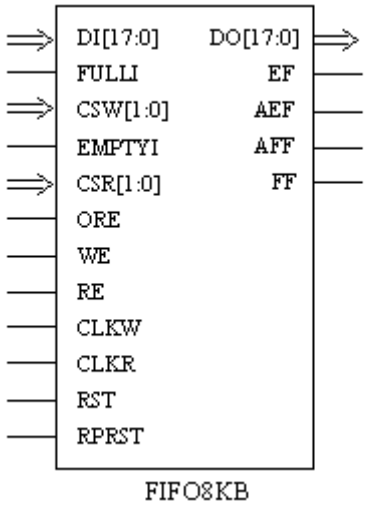
- TN1092 - MachXO Memory Usage Guide

FIFO8KB

8K FIFO Block RAM

Architecture Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: DI17, DI16, DI15, DI14, DI13, DI12, DI11, DI10, DI9, DI8, DI7, DI6, DI5, DI4, DI3, DI2, DI1, DI0, FULLI, CSW1, CSW0, EMPTYI, CSR1, CSR0, ORE, WE, RE, CLKW, CLKR, RST, RPRST

OUTPUTS: DO17, DO16, DO15, DO14, DO13, DO12, DO11, DO10, DO9, DO8, DO7, DO6, DO5, DO4, DO3, DO2, DO1, DO0, EF, AEF, AFF, FF

ATTRIBUTES:

DATA_WIDTH_W: 1, 2, 4, 9, 18 (default)

DATA_WIDTH_R: 1, 2, 4, 9, 18 (default)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 2-bit binary value (default: zeros)

CSDECODE_R: any 2-bit binary value (default: zeros)

GSR: "DISABLED" (default), "ENABLED"

RESETMODE: "ASYNC" (default), "SYNC"

ASYNC_RESET_RELEASE: "SYNC" (default), "ASYNC"

AEPOINTER: any 14-bit binary value (default: all zeros)

AEPOINTER1: any 14-bit binary value (default: all zeros)

AFPOINTER: any 14-bit binary value (default: all zeros)

AFPOINTER1: any 14-bit binary value (default: all zeros)

FULLPOINTER: any 14-bit binary value (default: all zeros)

FULLPOINTER1: any 14-bit binary value (default: all zeros)

Description

The following table describes the I/O ports of the FIFO8KB primitive.

Table 683:

Port Name	I/O	Definition
DI[17:0]	I	Write data (up to 18)
CLKW	I	Write clock
WE	I	Write clock enable
RST	I	Reset write pointers
FULLI	I	Chip select write
CSW[1:0]	I	Chip select write
CLKR	I	Read clock
RE	I	Read clock enable
ORE	I	Read output clock enable
EMPTYI	I	Chip select read
CSR[1:0]	I	Chip select read
RPRST	I	Reset read pointers
DO[17:0]	O	Read data (up to 18)
AFF	O	Almost full flag
FF	O	Full flag
AEF	O	Almost empty flag
EF	O	Empty flag

You can refer to the following technical note on the Lattice web site on detailed information and usage.

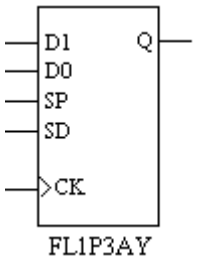
- TN1201 - Memory Usage Guide for MachXO2 Devices

FL1P3AY

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR Used for Preset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SP, CK, SD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 684:

INPUTS					OUTPUTS
D0	D1	SP	SD	CK	Q
X	X	0	X	X	Q
0	X	1	0	↑	0
1	X	1	0	↑	1
X	0	1	1	↑	0
X	1	1	1	↑	1

X = Don’t care

When GSR=0, Q=1 (D0=D1=SP=SD=CK=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

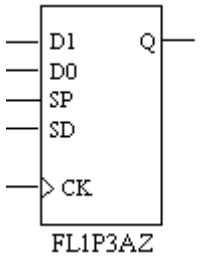
FL1P3AZ

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, and Positive Level Enable, GSR Used for Clear

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D

- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SP, CK, SD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 685:

INPUTS					OUTPUTS
D0	D1	SP	SD	CK	Q
X	X	0	X	X	Q
0	X	1	0	↑	0
1	X	1	0	↑	1
X	0	1	1	↑	0
X	1	1	1	↑	1

X = Don’t care

When GSR=0, Q=0 (D0=D1=SP=SD=CK=X)

Note

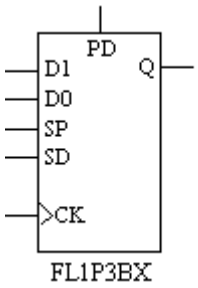
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1P3BX

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Preset, and Positive Level Enable

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SP, CK, SD, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 686:

INPUTS						OUTPUTS
D0	D1	SP	SD	CK	PD	Q
X	X	0	X	X	0	Q
X	X	X	X	X	1	1
0	X	1	0	↑	0	0
1	X	1	0	↑	X	1
X	0	1	1	↑	0	0
X	1	1	1	↑	X	1

X = Don’t care

When GSR=0, Q=1 (D0=D1=SP=SD=CK=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

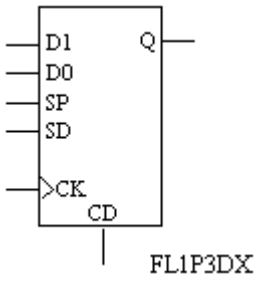
FL1P3DX

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Asynchronous Clear, and Positive Level Enable

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SP, CK, SD, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 687:

INPUTS						OUTPUTS
D0	D1	SP	SD	CK	CD	Q
X	X	0	X	X	0	Q
X	X	X	X	X	1	0
0	X	1	0	↑	X	0
1	X	1	0	↑	0	1
X	0	1	1	↑	X	0
X	1	1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D0=D1=SP=SD=CK=CD=X)

Note

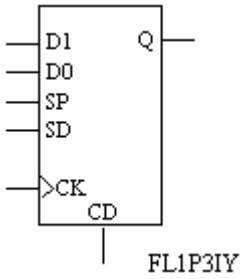
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1P3IY

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Clear, and Positive Level Enable (Clear overrides Enable)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SP, SD, CK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 688:

INPUTS						OUTPUTS
D0	D1	SP	SD	CK	CD	Q
X	X	0	X	X	0	Q
X	X	X	X	↑	1	0
0	X	1	0	↑	X	0
1	X	1	0	↑	0	1
X	0	1	1	↑	X	0
X	1	1	1	↑	0	1

X = Don’t care

When GSR=0, Q=1 (D0=D1=SP=SD=CK=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

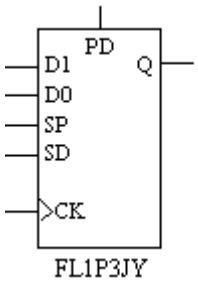
FL1P3JY

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, Positive Level Data Select, Positive Level Synchronous Preset, and Positive Level Enable (Preset overrides Enable)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SP, CK, SD, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 689:

INPUTS						OUTPUTS
D0	D1	SP	SD	CK	PD	Q
X	X	0	X	X	0	Q
X	X	X	X	↑	1	1
0	X	1	0	↑	0	0
1	X	1	0	↑	X	1
X	0	1	1	↑	0	0
X	1	1	1	↑	X	1

X = Don’t care

When GSR=0, Q=1 (D0=D1=SP=SD=CK=PD=X)

Note

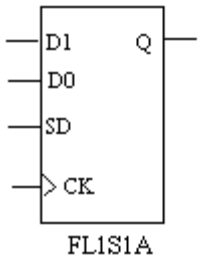
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1S1A

Positive Level Loadable Latch with Positive Select, GSR Used for Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 690:

INPUTS				OUTPUTS
D0	D1	SD	CK	Q
0	X	0	1	0
1	X	0	1	1
X	0	1	1	0
X	1	1	1	1

X = Don’t care

When GSR=0, Q=0 (D0=D1=SD=CK=X)

Note

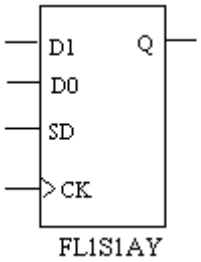
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1S1AY

Positive Level Loadable Latch with Positive Select, GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 691:

INPUTS				OUTPUTS
D0	D1	SD	CK	Q
0	X	0	1	0
1	X	0	1	1
X	0	1	1	0
X	1	1	1	1

X = Don’t care

When GSR=0, Q=1 (D0=D1=SD=CK=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

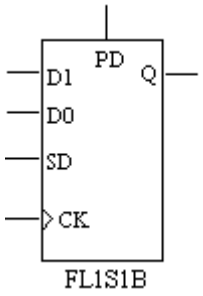
FL1S1B

Positive Level Loadable Latch with Positive Level Data Select and Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC

- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 692:

INPUTS					OUTPUTS
D0	D1	SD	CK	PD	Q
X	X	X	0	0	Q
X	X	X	X	1	1
0	X	0	1	0	0
1	X	0	1	X	1
X	0	1	1	0	0
X	1	1	1	X	1

X= Don’t care

When GSR=0, Q=1 (D0=D1=SD=CK=PD=X)

Note

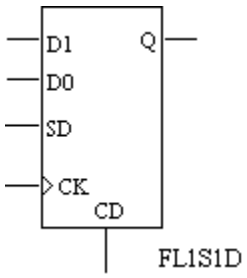
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1S1D

Positive Level Loadable Latch with Positive Level Data Select and Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 693:

INPUTS					OUTPUTS
D0	D1	SD	CK	CD	Q
X	X	X	0	0	Q
X	X	X	X	1	0
0	X	0	1	X	0
1	X	0	1	0	1
X	0	1	1	X	0
X	1	1	1	0	1

X = Don’t care

When GSR=0, Q=0 (D0=D1=SD=CK=CD=X)

Note

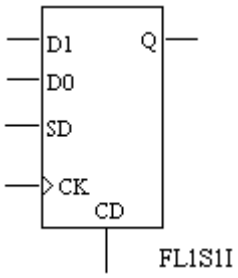
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1S1I

Positive Level Loadable Latch with Positive Level Data Select and Positive Level Synchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 694:

INPUTS					OUTPUTS
D0	D1	SD	CK	CD	Q
X	X	X	0	0	Q
X	X	X	1	1	0
0	X	0	1	X	0
1	X	0	1	0	1
X	0	1	1	X	0
X	1	1	1	0	1

X = Don’t care

When GSR=0, Q=0 (D0=D1=SD=CK=CD=X)

Note

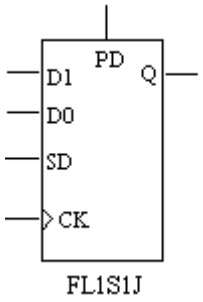
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1S1J

Positive Level Loadable Latch with Positive Level Data Select and Positive Level Synchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 695:

INPUTS					OUTPUTS
D0	D1	SD	CK	PD	Q
X	X	X	0	0	Q
X	X	X	1	1	1
0	X	0	1	0	0
1	X	0	1	X	1
X	0	1	1	0	0
X	1	1	1	X	1

X = Don’t care

When GSR=0, Q=1 (D0=D1=SD=CK=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

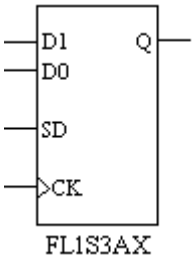
FL1S3AX

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR Used for Clear

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 696:

INPUTS				OUTPUTS
D0	D1	SD	CK	Q
0	X	0	↑	0
1	X	0	↑	1
X	0	1	↑	0
X	1	1	↑	1

X = Don’t care

When GSR=0, Q=0 (D0=D1=SD=CK=X)

Note

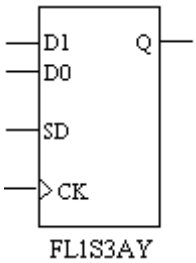
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FL1S3AY

Positive Edge Triggered D Flip-Flop with 2 Input Data Mux, GSR Used for Preset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CK, SD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 697:

INPUTS				OUTPUTS
D0	D1	SD	CK	Q
0	X	0	↑	0
1	X	0	↑	1
X	0	1	↑	0
X	1	1	↑	1

X = Don’t care

When GSR=0, Q=1 (D0=D1=SD=CK=X)

Note

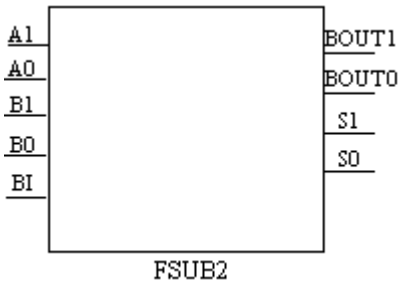
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FSUB2

2 Bit Fast Subtractor (two's complement)

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP
- MachXO
- Platform Manager



INPUTS: A1, A0, B1, B0, BI

OUTPUTS: BOUT1, BOUT0, S1, S0

Description

FSUB2 is a 2-bit two's complement subtractor. It has a borrow-in input (BI) and two 2-bit input (A0, A1 and B0, B1). The FSUB2 produces a 2-bit difference output (S0, S1) along with a 2-bit borrow-out output (BOUT1, BOUT).

Example pin functions:

Table 698:

Function	Pins
input	A1, A0, B1, B0
output	S1, S0
borrow-in input	BI
borrow-out output (Bit-0)	BOUT0
borrow-out output (Bit-1)	BOUT1

Truth Table

Table 699:

INPUTS					OUTPUTS			
A0	A1	B0	B1	BI	S0	BOUT0	S1	BOUT1
0	0	0	0	0	0	0	1	0
1	1	0	0	0	1	1	0	1
0	0	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1	0
0	0	0	0	1	1	1	0	1
1	1	0	0	1	1	1	1	1
0	0	1	1	1	0	0	1	0
1	1	1	1	1	1	1	0	1

Note

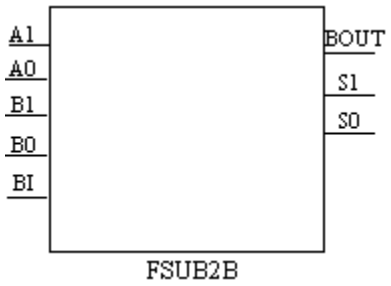
- BI and BO are inverse from standard two's complement behavior.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

FSUB2B

2 Bit Subtractor

Architectures Supported:

- LatticeECP2/M
- LatticeECP3
- LatticeXP2
- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: A0, A1, B0, B1, BI

OUTPUTS: BOUT, S0, S1

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

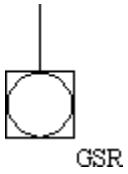
G

GSR

Global Set/Reset Interface

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: GSR

Description

GSR is used to reset or set all register elements in your design. The GSR component can be connected to a net from an input buffer or an internally generated net. It is active LOW and when pulsed will set or reset all flip-flops, latches, registers, and counters to the same state as the local set or reset functionality.

It is not necessary to connect signals for GSR to any register elements explicitly. The function will be implicitly connected globally. The functionality of the GSR for sequential cells without a local set or reset are described in the appropriate library help page.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

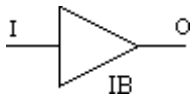
I

IB

Input Buffer

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: I

OUTPUT: O

Truth Table

Table 700:

INPUTS	OUTPUTS
I	O
1	1
0	0
Z	U

U = Unknown

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IBDDC

Dynamic Delay

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: I, DC3, DC2, DC1, DC0

OUTPUT: O

Description

The IBDDC primitive is used to control the input delay dynamically. See the below table for the I/O description.

Table 701:

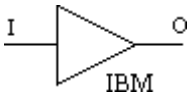
Port Name	I/O	Definition
I	Input	Input
DC[3:0]	Input	Dynamic delay control
O	Output	Output

IBM

CMOS Input Buffer

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUT: I

OUTPUT: O

Truth Table

Table 702:

INPUTS	OUTPUTS
I	O
1	1
0	0
Z	U

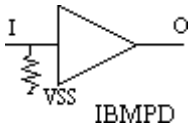
U = Unknown

IBMPD

CMOS Input Buffer with Pull-down

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUT: I

OUTPUT: O

Truth Table

Table 703:

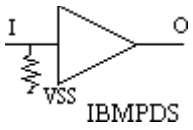
INPUTS	OUTPUTS
I	O
1	1
0	0
Z	0

IBMPDS

CMOS Input Buffer with Pull-down and Delay

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUT: I

OUTPUT: O

Truth Table

Table 704:

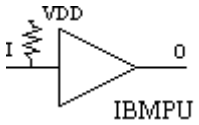
INPUTS	OUTPUTS
I	O
1	1
0	0
Z	0

IBMPU

CMOS Input Buffer with Pull-up

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUT: I

OUTPUT: O

Truth Table

Table 705:

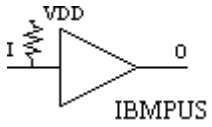
INPUTS	OUTPUTS
I	O
1	1
0	0
Z	1

IBMPUS

CMOS Input Buffer with Pull-up and Delay

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUT: I

OUTPUT: O

Truth Table

Table 706:

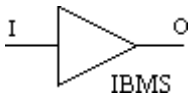
INPUTS	OUTPUTS
I	O
1	1
0	0
Z	1

IBMS

CMOS Input Buffer with Delay

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUT: I

OUTPUT: O

Truth Table

Table 707:

INPUTS	OUTPUTS
I	O
1	1
0	0
Z	U

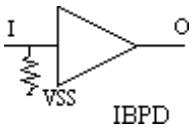
U = Unknown

IBPD

Input Buffer with Pull-down

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: I

OUTPUT: O

Truth Table

Table 708:

INPUTS	OUTPUTS
I	O
1	1
0	0
Z	0

Note

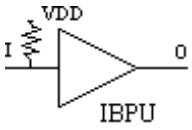
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IBPU

Input Buffer with Pull-up

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: I

OUTPUT: O

Truth Table

Table 709:

INPUTS	OUTPUTS
I	O
1	1
0	0
Z	1

Note

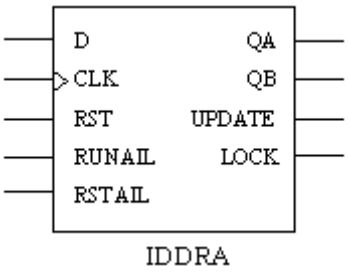
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IDDRA

Input DDR

Architectures Supported:

- LatticeSC/M

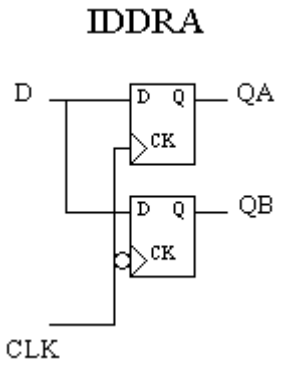


INPUTS: D, CLK, RST, RUNAIL, RSTAIL

OUTPUTS: QA, QB, UPDATE, LOCK

Description

Double Data Rate input logic. The following symbolic diagram shows the flip-flop structure of this primitive.



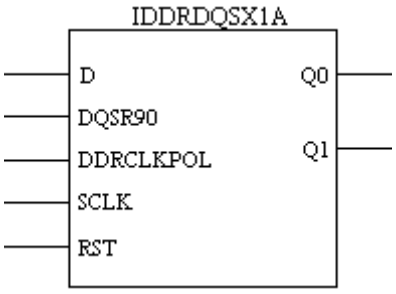
For more usage, see related technical notes or contact technical support.

IDDRDQSX1A

Input for DDR1/2 Memory

Architectures Supported:

- MachXO2
- Platform Manager 2



INPUTS: D, DQSR90, DDRCLKPOL, SCLK, RST

OUTPUTS: Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

IDDR supports two clock domains (right side only). IDDRDQSX1A is the input for DDR1/2 memory. It is used for right bank only.

See the following table for port description of the IDDRDQSX1A primitive.

Table 710:

Port Name	I/O	Definition
D	I	DDR input from sysIO buffer.
DQSR90	I	Shifted DQS input for read.
DDRCLKPOL	I	DQS clock polarity. This signal is used to connect to the DDRCLKPOL output of DQSBUFH.
SCLK	I	System clock.
RST	I	RESET to this block from CIB.
Q0	O	Data at the positive edge of the clock.
Q1	O	Data at the negative edge of the clock.

For more information and usage, refer to the following technical note on the Lattice web site.

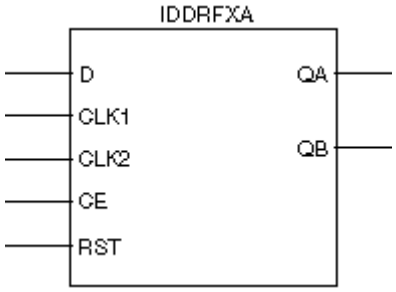
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

IDDRFXA

DDR Generic Input with Full Clock Transfer (x1 Gearbox)

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: D, CLK1, CLK2, RST, CE

OUTPUTS: QA, QB

Description

This primitive inputs DDR data at both edges of clock CLK1 and generates two streams of data aligned to clock CLK2. CLK1 is used to register the DDR registers and the first set of synchronization registers. CLK2 is used by the third stage of registers. Both CLK1 and CLK2 should be clocked by the FPGA clock. The LatticeECP2/M Family Data Sheet explains the input register block in more detail.

Note that LSR is only for second stage register/latch. For supporting DDR modes configured for bidirectional use, software will tie LSR LOW for input registers. The default for LSR is HIGH.

See the following table for port description of the IDDRFXA primitive.

Table 711:

Port Name	I/O	Definition
D	I	DDR data
CLK1	I	Clock connected to the FPGA clock
CLK2	I	Clock connected to the FPGA clock
CE	I	Clock enable signal
RST	I	Signal used to reset the DDR register
QA	O	Data at the positive edge of the CLK
QB	O	Data at the negative edge of the CLK

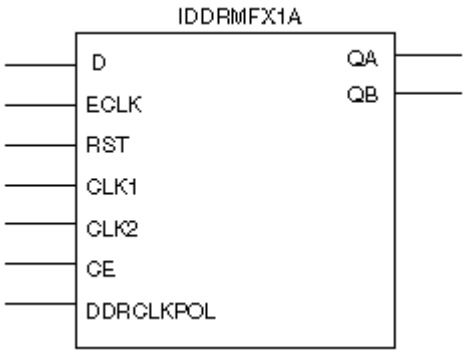
For more usage, see related technical notes or contact technical support.

IDDRMFX1A

DDR Input and DQS to System Clock Transfer Registers with Full Clock Cycle Transfer

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: D, ECLK, CLK1, CLK2, RST, CE, DDRCLKPOL

OUTPUTS: QA, QB

Description

This primitive implements a full clock cycle transfer as compared the to the IDDRMX1A primitive that will only implement a half clock cycle transfer. The DDR registers are designed to use edge clock routing on the I/O side and the primary clock on the FPGA side. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFC primitive). The CLK1 and CLK2 inputs should be connected to the slow system (FPGA) clock. DDRCLKPOL is an input from the DQS Clock Polarity tree. This signal is generated by the DQS Transition detect circuit in the hardware.

Note that the DDRCLKPOL input to IDDRMFX1A should be connected to the DDRCLKPOL output of DQSBUFC. LSR is only for second stage register/ latch. For supporting DDR modes configured for bidirectional use, software will tie LSR LOW for input registers. The default for LSR is HIGH.

See the following table for port description of the IDDRMFX1A primitive.

Table 712:

Port Name	I/O	Definition
D	I	DDR data.
ECLK	I	The phase shifted DQS should be connected to this input.
RST	I	Reset.
CLK1	I	Slow FPGA CLK.
CLK2	I	Slow FPGA CLK.
CE	I	Clock enable.
DDRCLKPOL	I	DDR clock polarity signal.
QA	O	Data at the positive edge of the CLK.
QB	O	Data at the negative edge of the CLK.

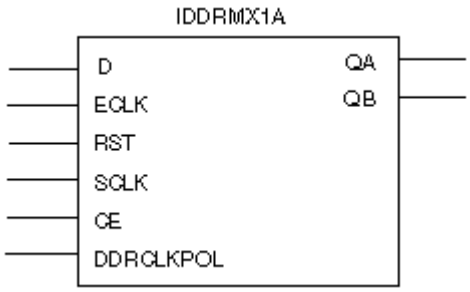
For more usage, see related technical notes or contact technical support.

IDDRMX1A

DDR Input and DQS to System Clock Transfer Registers with Half Clock Cycle Transfer

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: D, ECLK, SCLK, RST, CE, DDRCLKPOL

OUTPUTS: QA, QB

Description

This primitive implements the input register block. The DDR registers are designed to use edge clock routing on the I/O side and the primary clock on the FPGA side. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFC primitive). The SCLK input should be connected to the system (FPGA) clock. DDRCLKPOL is an input from the DQS Clock Polarity tree. This signal is generated by the DQS Transition detect circuit in the hardware. The DDRCLKPOL signal is used to choose the polarity of the SCLK to the synchronization registers.

Note that the DDRCLKPOL input to IDDRMX1A should be connected to the DDRCLKPOL output of DQSBUFC. LSR is only for second stage register/ latch. For supporting DDR modes configured for bidirectional use, software will tie LSR LOW for input registers. The default for LSR is HIGH.

See the following table for port description of the IDDRMX1A primitive.

Table 713:

Port Name	I/O	Definition
D	I	DDR data.
ECLK	I	The phase shifted DQS should be connected to this input.

Table 713:

Port Name	I/O	Definition
RST	I	Reset.
SCLK	I	System CLK.
CE	I	Clock enable.
DDRCLKPOL	I	DDR clock polarity signal.
QA	O	Data at the positive edge of the CLK.
QB	O	Data at the negative edge of the CLK.

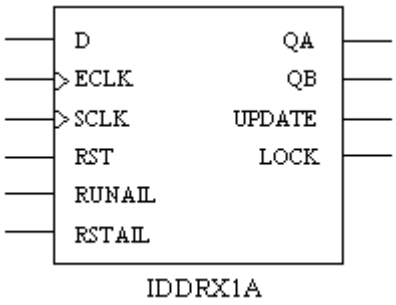
For more usage, see related technical notes or contact technical support.

IDDRX1A

Input DDR

Architectures Supported:

- LatticeSC/M

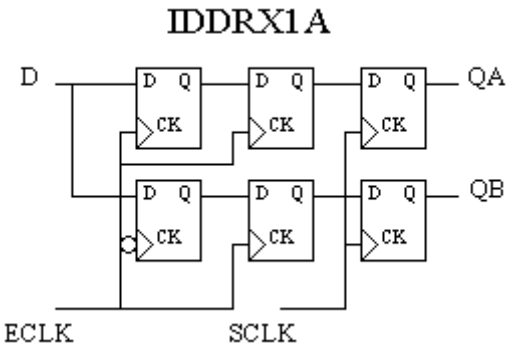


INPUTS: D, ECLK, SCLK, RST, RUNAIL, RSTAIL

OUTPUTS: QA, QB, UPDATE, LOCK

Description

Double Data Rate input logic. The input register block captures DDR input data using edge clock to primary clock domain transfer. It can also be set to perform the same functions as in the shift mode. The following symbolic diagram shows the flip-flop structure of this primitive.



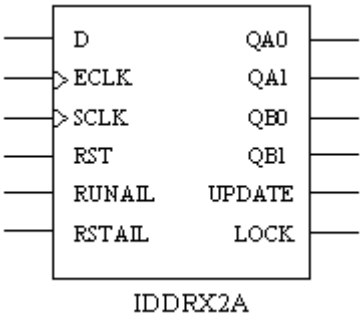
For more usage, see related technical notes or contact technical support.

IDDRX2A

Input DDR

Architectures Supported:

- LatticeSC/M

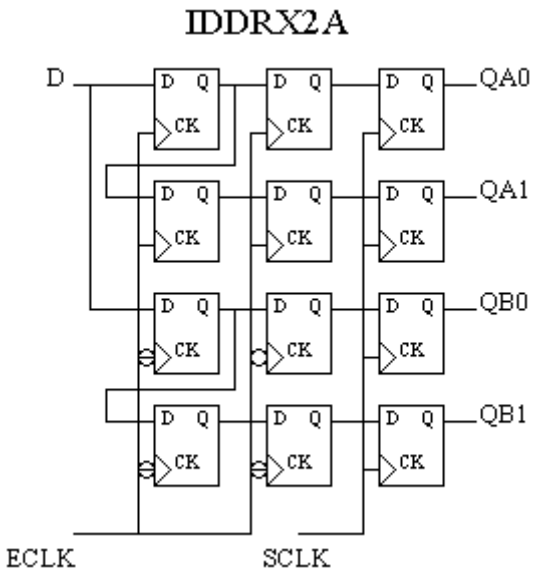


INPUTS: D, ECLK, SCLK, RST, RUNAIL, RSTAIL

OUTPUTS: QA0, QA1, QB0, QB1, UPDATE, LOCK

Description

Double Data Rate input logic. The input register block captures DDR input data using edge clock to primary clock domain transfer. It can also be set to perform the same functions as in the shift mode. The following symbolic diagram shows the flip-flop structure of this primitive.



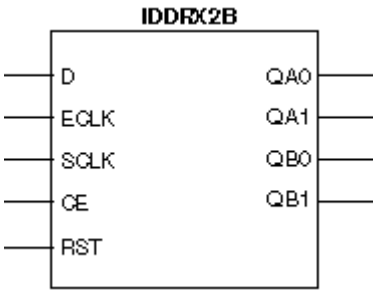
For more usage, see related technical notes or contact technical support.

IDDRX2B

DDR Generic Input with 2x Gearing Ratio

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: D, ECLK, SCLK, CE, RST

OUTPUTS: QA0, QA1, QB0, QB1

Description

This primitive is used when a gearing function is required. This primitive inputs DDR data at both edges of the edge and generates four streams of data. The DDR registers and the first set of synchronization registers are clocked by the ECLK input of the primitive, which should be connected to the fast ECLK. SCLK is used to clock the third stage of registers and should be connected to the FPGA clock. This primitive outputs four streams of data. Two of these data streams are generated using the complementary PIO registers.

Note that LSR is only for second stage register/latch. For supporting DDR modes configured for bidirectional use, software will tie LSR LOW for input registers. The default for LSR is HIGH.

See the following table for port description of the IDDRX2B primitive.

IDDRX2B Ports

Table 714:

Port Name	I/O	Definition
D	I	DDR data
ECLK	I	Clock connected to the fast edge clock
SCLK	I	Clock connected to the FPGA clock
CE	I	Clock enable signal
RST	I	Signal used to reset the DDR register
QA0, QA1	O	Data at the positive edge of the CLK
QB0, QB1	O	Data at the negative edge of the CLK

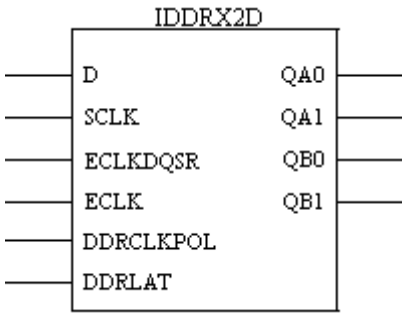
For more usage, see related technical notes or contact technical support.

IDDRX2D

Input DDR for DDR3_MEM, DDR_GENX2, and DDR3_MEMGEN

Architectures Supported:

- LatticeECP3



INPUTS: D, SCLK, ECLKDQSR, ECLK, DDRLAT, DDRCLKPOL

OUTPUTS: QA0, QA1, QB0, QB1

ATTRIBUTES:

(*EA only*) SCLKLATENCY: 1 (default), 2

Description

IDDRX2D is the input DDR for DDR3_MEM, DDR_GENX2 (DDR generic mode in X2 gearing), and DDR3_MEMGEN.

- E: DDR_GENX2 (left/right/top)
- E and EA: DDR3_MEM (left/right)
- E and EA: DDR3_MEMGEN (left/right/top)

See the below table for its port description.

Table 715:

Signal	I/O	Description
D	I	DDR input from sysIO buffer.
ECLK	I	Edge clock. Goes to the second stage of DDR registers.
SCLK	I	System clock. Clock used to transfer from the ECLK to the SCLK domain. SCLK = 1/2 ECLK rate. Goes to the third stage of registers.
ECLKDQSR	I	Phase shifted DQS in case of DDR memory interface. Edge clock for generic DDR interfaces. Connects to DQSBUF. For EA devices, ECLKDQSR should be used only for the DQS strobe.
DDRCLKPOL	I	DDR clock polarity signal.
DDRLAT	I	DDR latch control to input logic. Used to guarantee IDDRX2 gearing by selectively enabling a D flip-flop in the data path.
QA0	O	Data at the positive edge of the clock (IPA).

Table 715:

Signal	I/O	Description
QA1	O	Data at the positive edge of the clock (IPB).
QB0	O	Data at the negative edge of the clock (INA).
QB1	O	Data at the negative edge of the clock (INB).

For more information and usage, refer to the following technical note on the Lattice web site.

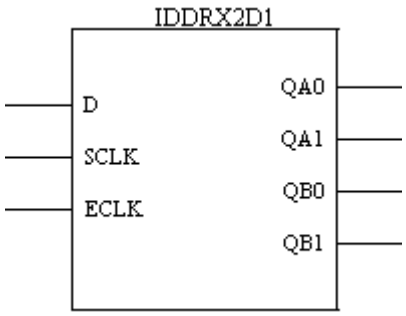
- TN1177 - LatticeECP3 sysIO Usage Guide

IDDRX2D1

Input DDR for DDR_GENX2

Architectures Supported:

- LatticeECP3



INPUTS: D, SCLK, ECLK

OUTPUTS: QA0, QA1, QB0, QB1

ATTRIBUTES:

(*EA only*) DR_CONFIG: "DISABLED" (default), "ENABLED"

Description

IDDRX2D1 is the input DDR for DDR_GENX2 (DDR generic mode in X2 gearing).

- EA: DDR_GENX2 (left/right/top)

See the below table for its port description.

Table 716:

Signal	I/O	Description
D	I	DDR input from sysIO buffer.
ECLK	I	Edge clock. Goes to the first and second stage of DDR registers.
SCLK	I	System clock. Clock used to transfer from the ECLK to the SCLK domain. SCLK = 1/2 ECLK rate. Goes to the third stage of registers.
QA0	O	Data at the positive edge of the clock (IPA).
QA1	O	Data at the positive edge of the clock (IPB).
QB0	O	Data at the negative edge of the clock (INA).
QB1	O	Data at the negative edge of the clock (INB).

For more information and usage, refer to the following technical note on the Lattice web site.

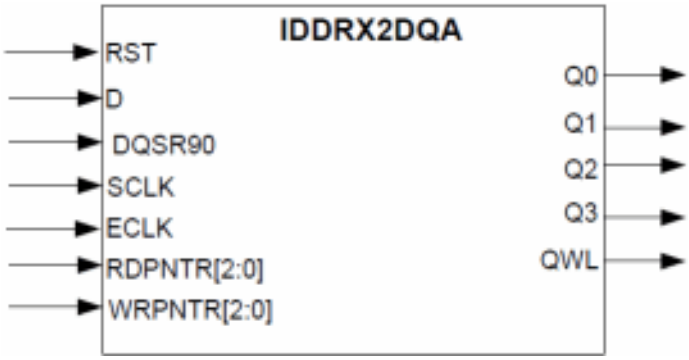
- TN1177 - LatticeECP3 sysIO Usage Guide

IDDRX2DQA

This primitive is used to implement DDR2 memory input interface at higher speeds and DDR3 memory interface.

Architectures Supported:

- ECP5



INPUTS: RST, D, DQSR90, SCLK, ECLK, RDPNTR2, RDPNTR1, RDPNTR0, WRPNTR2, WRPNTR1, WRPNTR0,

OUTPUTS: Q0, Q1, Q3, QWL

See the following table for port description of the IDDRX1DQA primitive.

Table 717:

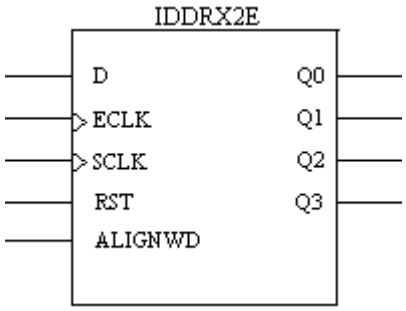
Port Name	I/O	Definition
D	I	DDR Data input
RST	I	Reset to DDR registers
DQSR90	I	DQS clock Input
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide by 2 of ECLK)
RDPNTR[2:0]	I	Read Pointer from the DQSBUF module used to transfer data to ECLK
WRPNTR[2:0]	I	Write Pointer from the DQSBUF module used to transfer data to ECLK
Q0, Q2	O	Data at positive edge of DQS
Q1, Q3	O	Data at negative edge of DQS
QWL	O	Data output used for Write Leveling

IDDRX2E

Input for Generic DDR X2 Using 1:4 Gearing

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, ECLK, SCLK, RST, ALIGNWD

OUTPUTS: Q0, Q1, Q2, Q3

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

IDDRX2E is the input for generic DDR X2 using 1:4 gearing. It uses the VPIC_RX hardware cell. It is used for bottom bank only.

See the below table for port description.

Table 718:

Signal	I/O	Description
D	I	DDR input from sysIO buffer.
ECLK	I	Clock connected to the high speed edge clock tree.
SCLK	I	Clock connected to the system clock.
RST	I	RESET to this block from CIB.
ALIGNWD	I	Data alignment signal used for word alignment. Each operation shifts word alignment by 1 bit.
Q0, Q2	O	Data available at the same edge of the clock.
Q1, Q3	O	Data available at the same edge of the clock.

For more information and usage, refer to the following technical note on the Lattice web site.

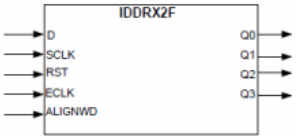
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

IDDRX2F

Generic input DDR Primitive

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: D, SCLK, RST, ECLK, ALIGNWD

OUTPUTS: Q0, Q1, Q2, Q3

Description

This primitive is used for Generic X1 IDDR implementation. The following table gives the port description of the IDDRX2F primitive.

Table 719:

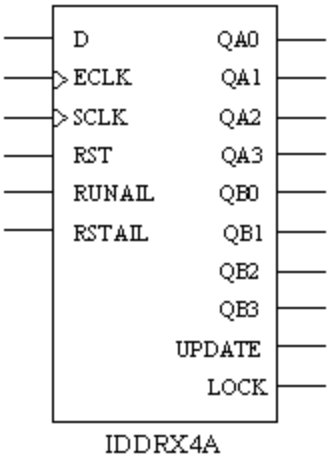
Port Name	I/O	Definition
D	I	DDR Data input.
SCLK	I	Primary clock input (divide by 2 of ECLK).
RST	I	Reset to DDR registers.
ECLK	I	Fast Edge clock.
ALIGNWD	I	This signal is used for Word alignment. It will shift word by one bit.
Q0, Q2	O	Data at positive edge of input ECLK.
Q1, Q3	O	Data at negative edge of input ECLK.

IDDRX4A

Input DDR

Architectures Supported:

- LatticeSC/M

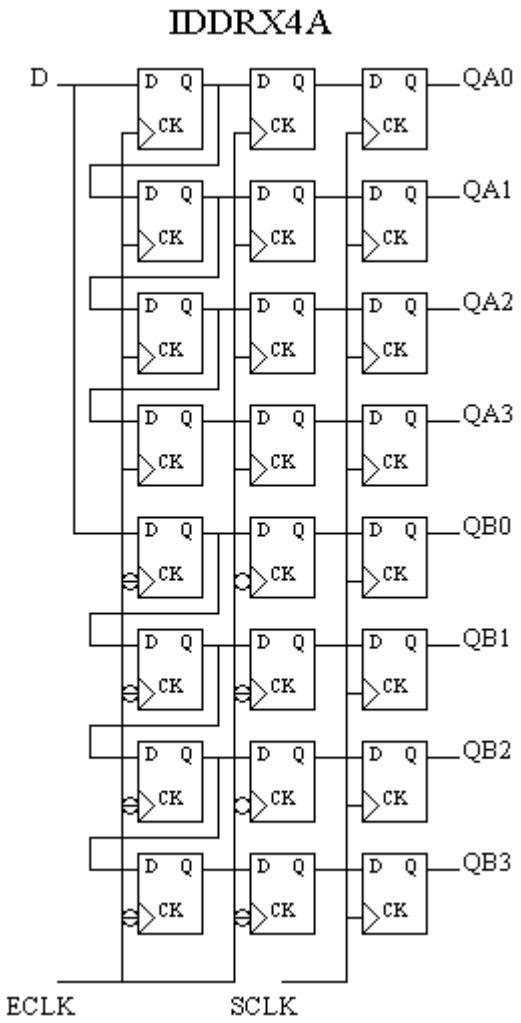


INPUTS: D, ECLK, SCLK, RST, RUNAIL, RSTAIL

OUTPUTS: QA0, QA1, QA2, QA3, QB0, QB1, QB2, QB3, UPDATE, LOCK

Description

Double Data Rate input logic. The input register block captures DDR input data using edge clock to primary clock domain transfer. It can also be set to perform the same functions as in the shift mode. The following symbolic diagram shows the flip-flop structure of this primitive.



For more information, see related technical notes or contact technical support.

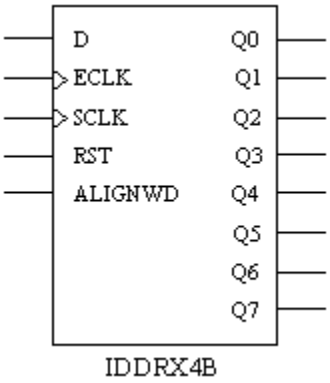
IDDRX4B

Input for Generic DDR X4 Using 1:8 Gearing

Architectures Supported:

- LFMNX
- MachXO2

- MachXO3D
- Platform Manager 2



INPUTS: D, ECLK, SCLK, RST, ALIGNWD

OUTPUTS: Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

IDDRX4B is the input for generic DDR X4 using 1:8 gearing. It uses the VPIC_RX hardware cell. It is used for bottom bank only.

See the below table for IDDRX4B I/O description.

Table 720:

Signal	I/O	Description
D	I	DDR input from sysIO buffer.
ECLK	I	Clock connected to the high speed edge clock tree.
SCLK	I	Clock connected to the system clock.
RST	I	RESET to this block from CIB.
ALIGNWD	I	Data alignment signal used for word alignment. Each operation shifts alignment by 1 bit.
Q0, Q2, Q4, Q6	O	Data available at the same edge of the clock.
Q1, Q3, Q5, Q7	O	Data available at the same edge of the clock.

For more information and usage, refer to the following technical note on the Lattice web site.

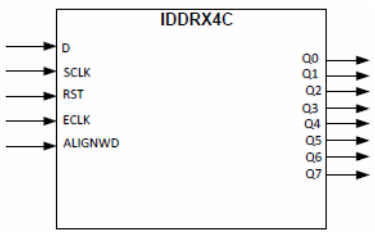
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

IDDRX4C

Input for Generic DDR X4 Using 1:8 Gearing

Architectures Supported:

- LIFMD
- LIFMDF



Input: D, ECLK, SCLK, RST, ALIGNWD;

Output: Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7;

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive is used for 8:1 Input side implementation.

See the below table for IDDRX4C I/O description.

Table 721:

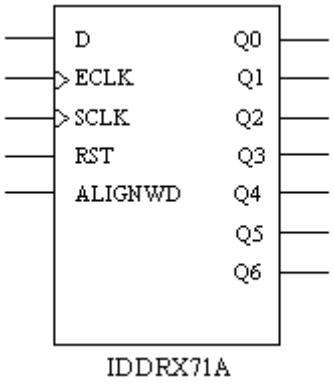
Port	I/O	Description
D	i	DDR Data input
ECLK	I	Edge clock
SCLK	I	Primary clock (Divide by 4 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for Word alignment. It will shift word by 1 bit.
Q0 to Q7	O	8 bits of output data

IDDRX71A

7:1 LVDS Input Supporting 1:7 Gearing

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, ECLK, SCLK, RST, ALIGNWD

OUTPUTS: Q0, Q1, Q2, Q3, Q4, Q5, Q6

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

IDDRX71A is the 7:1 LVDS input supporting 1:7 gearing. It is used for bottom bank only. IDDRX71A includes the SLIP circuitry.

See the below table for IDDRX71A I/O description.

Table 722:

Signal	I/O	Description
D	I	DDR input from sysIO buffer.
ECLK	I	Edge clock.
SCLK	I	Clock connected to the system clock.
RST	I	RESET for this block.
ALIGNWD	I	Data alignment signal used for 7:1 LVDS. Each operation shifts alignment by 2 bits.
Q0, Q1, Q2, Q3, Q4, Q5, Q6	O	1:7 LVDS output signals.

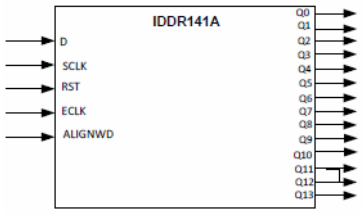
For more information and usage, refer to the following technical note on the Lattice web site.

- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

IDDR141A

Architectures Supported:

- LIFMD
- LIFMDF



Input: D, ECLK, SCLK, RST, ALIGNWD;

Output: Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12, Q13;

Description

This primitive is used for 14:1 Input side implementation. This is an extension of 7:1 for higher speeds.

See the below table for IDDR141A I/O description.

Table 723:

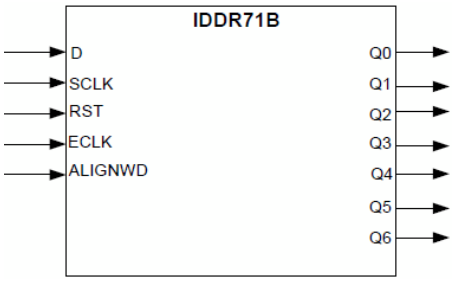
Port	I/O	Description
D	i	DDR Data input
ECLK	I	Edge clock
SCLK	I	Primary clock (Divide by 7 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for Word alignment. It will shift word by 1 bit.
Q0 to Q13	O	14 bits of output data

IDDR71B

7:1 LVDS Input Supporting 1:7 Gearing

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: D, ECLK, SCLK, RST, ALIGNWD

OUTPUTS: Q0, Q1, Q2, Q3, Q4, Q5, Q6

Description

This primitive is used for 7:1 LVDS Input side implementation

See the below table for IDDR71B I/O description.

Table 724:

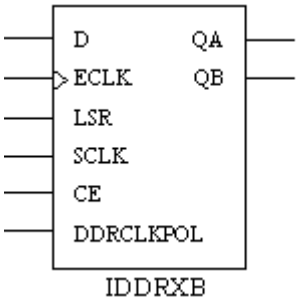
Signal	I/O	Description
D	I	DDR data input
ECLK	I	Edge clock.
SCLK	I	Primary clock. (Divide by 3.5 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for word alignment. It will shift word by 1 bit.
Q0, Q1, Q2, Q3, Q4, Q5, Q6	O	7 bits of output data.

IDDRXB

Input DDR

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: D, ECLK, LSR, SCLK, CE, DDRCLKPOL

OUTPUTS: QA, QB

ATTRIBUTES:

REGSET: "RESET" (default), "SET"

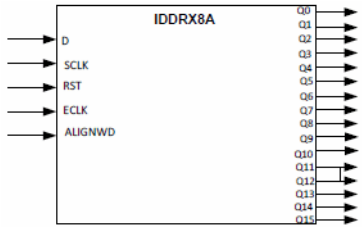
Description

Double Data Rate input logic with half cycle clock domain transfer for the negative edge captured data (both edges of captured data enter core with positive edge flip-flops. For more information, see related technical notes or contact technical support.

IDDRX8A

Architectures Supported:

- LIFMD
- LIFMDF



Input: D, ECLK, SCLK, RST, ALIGNWD;

Output: Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12, Q13;

Description

This primitive is used for 16:1 Input side implementation.

See the below table for IDDRX8A I/O description.

Table 725:

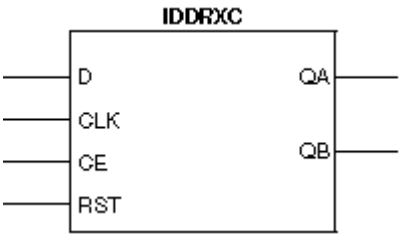
Port	I/O	Description
D	I	DDR Data input
ECLK	I	Edge clock
SCLK	I	Primary clock (Divide by 8 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for Word alignment. It will shift word by 1 bit.
Q0 to Q15	O	16 bits of output data

IDDRXC

DDR Generic Input

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: D, CLK, RST, CE

OUTPUTS: QA, QB

Description

This primitive inputs the DDR data at both edges of the edge and generates two streams of data. CLK of this module can be connected to either the edge clock or the primary FPGA clock.

Note that the DDRCLKPOL input to IDDRXC should be connected to the DDRCLKPOL output of DQSBUFC. LSR is only for second stage register/ latch. To support DDR modes configured for bidirectional use, software will tie LSR LOW for input registers. The default for LSR is HIGH.

See the following table for port description of the IDDRXC primitive.

Table 726:

Port Name	I/O	Definition
D	I	DDR input from sysIO buffer
CLK	I	Clock from the CIB
RST	I	RESET to this block from CIB
CE	I	Clock enable signal
QA	O	Data at the positive edge of the CLK
QB	O	Data at the negative edge of the CLK

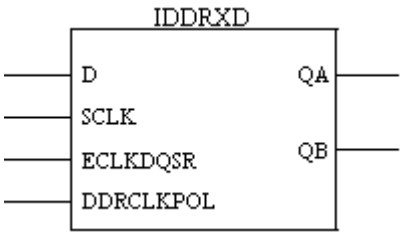
For more usage, see related technical notes or contact technical support.

IDDRXD

Input DDR for DDR_MEM, DDR2_MEM, DDR_GENX1, and DDR2_MEMGEN

Architectures Supported:

- LatticeECP3



INPUTS: D, SCLK, ECLKDQSR, DDRCLKPOL

OUTPUTS: QA, QB

ATTRIBUTES:

(*EA only*) SCLKLATENCY: 1 (default), 2

Description

IDDRXD is the input DDR for DDR_MEM, DDR2_MEM, DDR_GENX1 (DDR generic mode in X1 gearing), and DDR2_MEMGEN.

- E: DDR_MEM, DDR2_MEM, DDR_GENX1, and DDR2_MEMGEN (left/right/top)
- EA: DDR_MEM, DDR2_MEM, and DDR2_MEMGEN (left/right/top)

See the below table for its port description.

Table 727:

Signal	I/O	Description
D	I	DDR input from sysIO buffer.
SCLK	I	System clock.
ECLKDQSR	I	Phase shifted DQS in case of DDR memory interface. Connects to DQSBUF. For EA devices, ECLKDQSR should be used only for the DQS strobe.
DDRCLKPOL	I	DDR clock polarity signal.
QA	O	Data at the positive edge of the clock (mapped to IPB).
QB	O	Data at the negative edge of the clock (mapped to INB).

For more information and usage, refer to the following technical note on the Lattice web site.

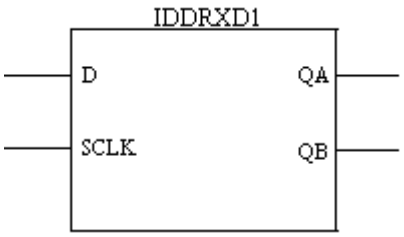
- TN1177 - LatticeECP3 sysIO Usage Guide

IDDRXD1

Input DDR for DDR_GENX1

Architectures Supported:

- LatticeECP3



INPUTS: D, SCLK

OUTPUTS: QA, QB

Description

IDDRXD1 is the input DDR for DDR_GENX1 (DDR generic mode in X1 gearing).

- EA: DDR_GENX1 (left/right/top)

See the below table for its port description.

Table 728:

Signal	I/O	Description
D	I	DDR input from sysIO buffer.
SCLK	I	System clock.
QA	O	Data at the positive edge of the clock (mapped to IPB).
QB	O	Data at the negative edge of the clock (mapped to INB).

For more information and usage, refer to the following technical note on the Lattice web site.

- TN1177 - LatticeECP3 sysIO Usage Guide

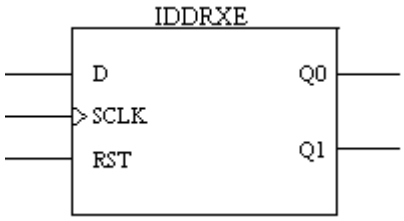
IDDRXE

Input for Generic DDR X1 Using 1:2 Gearing

Architectures Supported:

- LFMNX

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SCLK, RST

OUTPUTS: Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

IDDRXE is the input for generic DDR X1 using 1:2 gearing. It uses the mPIC (base pic) or PIC (memory pic) hardware cell. It is used for all sides.

See the below table for port description.

Table 729:

Signal	I/O	Description
D	I	DDR input from sysIO buffer
SCLK	I	Clock connected to the system clock
RST	I	RESET for this block
Q0	O	Data at the positive edge of the clock
Q1	O	Data at the negative edge of the clock

For more information and usage, refer to the following technical note on the Lattice web site.

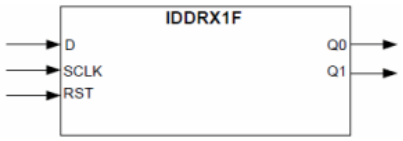
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

IDDRX1F

Generic input DDR Primitive

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: D, SCLK, RST

OUTPUTS: Q0, Q1

Description

- This primitive is used for Generic X1 IDDR implementation. The following table gives the port description of the IDDRX1F primitive.

Table 730:

Port Name	I/O	Definition
D	I	DDR Data input.
SCLK	I	Primary clock input.
RST	I	Reset to DDR registers.
Q0	O	Data at positive edge of clock.
Q1	O	Data at negative edge of clock.

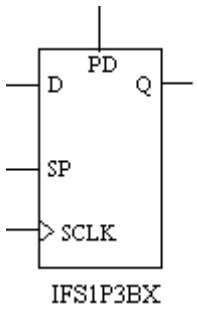
IFS1P3BX

Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX

- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SP, SCLK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 731:

INPUTS				OUTPUTS
D	SP	SCLK	PD	Q
X	0	X	0	Q
X	X	X	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=1 (D=SP=SCLK=PD=X)

Note

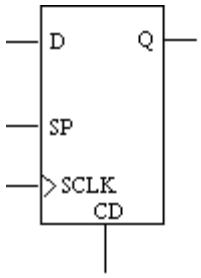
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IFS1P3DX

Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



IFS1P3DX

INPUTS: D, SP, SCLK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 732:

INPUTS				OUTPUTS
D	SP	SCLK	CD	Q
X	0	X	0	Q
X	X	X	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=SP=SCLK=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

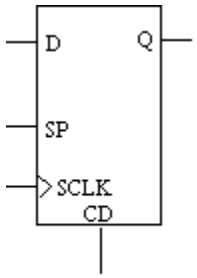
IFS1P3IX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable, and System Clock (Clear overrides Enable) (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2

- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



IFS1P3IX

INPUTS: D, SP, SCLK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 733:

INPUTS				OUTPUTS
D	SP	SCLK	CD	Q
X	0	X	0	Q
X	X	↑	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=SP=SCLK=CD=X)

Note

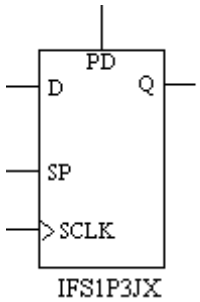
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IFS1P3JX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable, and System Clock (Preset overrides Enable) (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SP, SCLK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 734:

INPUTS				OUTPUTS
D	SP	SCLK	PD	Q
X	0	X	0	Q
X	X	↑	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=1 (D=SP=SCLK=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

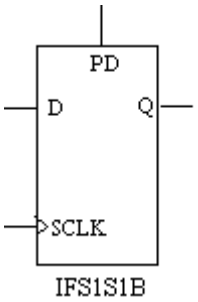
IFS1S1B

Positive Level Data Latch with Positive Level Asynchronous Preset and System Clock (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2

- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SCLK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 735:

INPUTS			OUTPUTS
D	SCLK	PD	Q
X	0	0	Q
X	X	1	1
0	1	0	0
1	1	0	1

X= Don’t care

When GSR=0, Q=1 (D=SCLK=PD=X)

Note

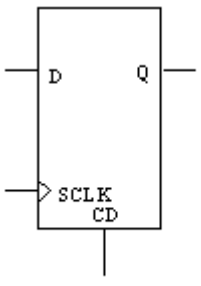
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IFS1S1D

Positive Level Data Latch with Positive Level Asynchronous Clear and System Clock (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



IFS1S1D

INPUTS: D, SCLK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 736:

INPUTS			OUTPUTS
D	SCLK	CD	Q
X	0	0	Q
X	X	1	0
0	1	0	0
1	1	0	1

X = Don’t care

When GSR=0, Q=0 (D=SCLK=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

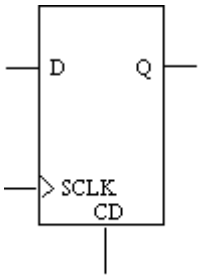
IFS1S1I

Positive Level Data Latch with Positive Level Synchronous Clear and System Clock (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2

- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



IFS1S1I

INPUTS: D, SCLK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 737:

INPUTS			OUTPUTS
D	SCLK	CD	Q
X	0	0	Q
X	1	1	0
0	1	0	0
1	1	0	1

X = Don’t care

When GSR=0, Q=0 (D=SCLK=CD=X)

Note

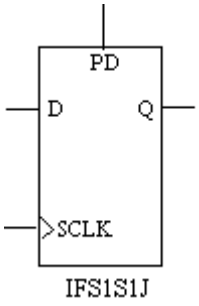
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IFS1S1J

Positive Level Data Latch with Positive Level Synchronous Preset and System Clock (used in input PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SCLK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 738:

INPUTS			OUTPUTS
D	SCLK	PD	Q
X	0	0	Q
X	1	1	1
0	1	0	0
1	1	0	1

X = Don’t care

When GSR=0, Q=1 (D=SCLK=PD=X)

Note

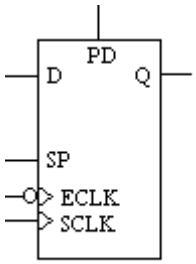
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

ILF2P3BX

Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, and Positive Level Asynchronous Preset (used in input PIC area only)

Architectures Supported:

- LatticeSC/M



ILF2P3BX

INPUTS: D, SP, ECLK, SCLK, PD

OUTPUT: Q

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 739:

INPUTS					OUTPUTS	
D	SP	ECLK	SCLK	PD	LATCH_Q	Q
X	X	X	X	1	LATCH_Q	1
X	0	1	X	0	LATCH_Q	Q
0	X	0	B	0	0	Q
1	X	0	B	0	1	Q
X	1	1	↑	0	0	0
X	1	1	↑	0	1	1
0	1	0	↑	0	0	0
1	1	0	↑	0	1	1

X = Don’t care

LATCH_Q = Output data from latch

B = Not rising edge

When GSR=0, Q=1 (D=SP=ECLK=SCLK=PD=X)

Note

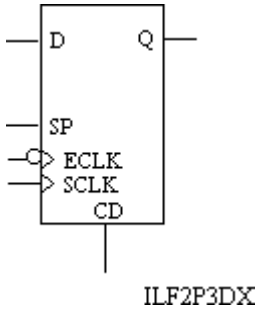
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

ILF2P3DX

Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, and Positive Level Asynchronous Clear (used in input PIC area only)

Architectures Supported:

- LatticeSC/M



INPUTS: D, SP, ECLK, SCLK, CD

OUTPUT: Q

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 740:

INPUTS					OUTPUTS	
D	SP	ECLK	SCLK	CD	LATCH_Q	Q
X	X	X	X	1	LATCH_Q	0
X	0	1	X	0	LATCH_Q	Q
0	X	0	B	0	0	Q
1	X	0	B	0	1	Q
X	1	1	↑	0	0	0
X	1	1	↑	0	1	1
0	1	0	↑	0	0	0
1	1	0	↑	0	1	1

X = Don’t care

LATCH_Q = Output data from latch

B = Not rising edge

When GSR=0, Q=0 (D=SP=ECLK=SCLK=CD=X)

Note

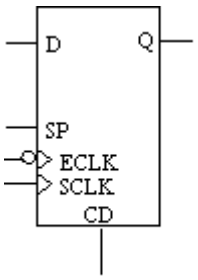
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

ILF2P3IX

Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable) (used in input PIC area only)

Architectures Supported:

- LatticeSC/M



ILF2P3IX

INPUTS: D, SP, ECLK, SCLK, CD

OUTPUT: Q

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 741:

INPUTS					OUTPUTS	
D	SP	ECLK	SCLK	CD	LATCH_Q	Q
X	X	X	↑	1	LATCH_Q	0
X	0	1	X	0	LATCH_Q	Q
0	X	0	B	0	0	Q
1	X	0	B	0	1	Q
X	1	1	↑	0	0	0
X	1	1	↑	0	1	1
0	1	0	↑	0	0	0
1	1	0	↑	0	1	1

X = Don’t care

LATCH_Q = Output data from latch

B = Not rising edge

When GSR = 0, Q = 0 (D = SP = ECLK = SCLK = CD = X)

Note

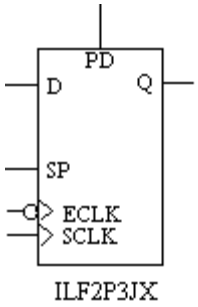
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

ILF2P3JX

Negative Level Edge Clocked (ECLK) Latch, Feeding Positive Edge Triggered System Clocked (SCLK) Flip-Flop, and Positive Level Synchronous Preset (Preset overrides Enable) (used in input PIC area only)

Architectures Supported:

- LatticeSC/M



INPUTS: D, SP, ECLK, SCLK, PD

OUTPUT: Q

Description

This primitive must be paired with an input or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 742:

INPUTS					OUTPUTS	
D	SP	ECLK	SCLK	PD	LATCH_Q	Q
X	X	X	↑	1	LATCH_Q	1
X	0	1	X	0	LATCH_Q	Q
0	X	0	B	0	0	Q
1	X	0	B	0	1	Q
X	1	1	↑	0	0	0
X	1	1	↑	0	1	1
0	1	0	↑	0	0	0
1	1	0	↑	0	1	1

X = Don’t care

LATCH_Q = Output data from latch

B = Not rising edge

When GSR = 0, Q = 1 (D = SP = ECLK = SCLK = PD = X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

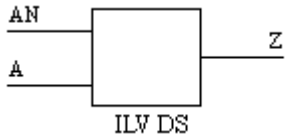
ILVDS

LVDS Input Buffer

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD

- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, AN

OUTPUT: Z

Truth Table

Table 743:

INPUTS		OUTPUTS
A	AN	Z
0	1	0
1	0	1

Note

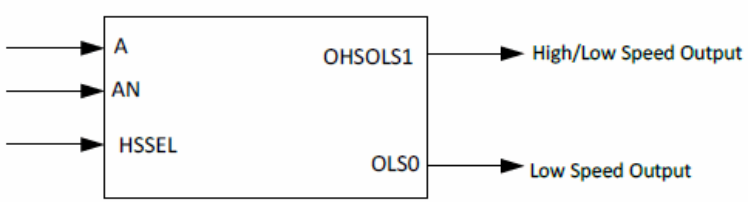
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IMIPI

Special Primitive for MIPI Input Support

Architectures Supported:

- ECP5



INPUTS:A, AN, HSSEL

OUTPUT: OHSLS, OLS

The IMIPI I/O description is shown below.

Table 744:

Port Name	I/O	Description
A	I	PAD C Input
AN	I	PAD D Input
HSSEL	I	High Speed Select Signal. This is shared with the Tristate input of the buffer. HSSEL=1: High Speed mode, 100 ohm differential termination is on. PADC logic select differential signal to IOL for gearing. HS_SEL=0: Low Speed mode, 100 ohm termination is turned off. OHSLS selected as ratioed lvcmos input buffer from I input (PADC), OLS selected as lvcmos input from IN input (PADD).
OHSLS	O	High Speed or Low Speed Output depending on HSSEL
OLS	O	Low speed output.

Description

Primitive used when implementing MIPI interface.

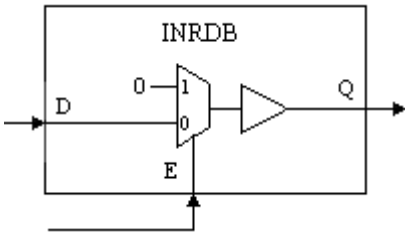
INRDB

Input Reference and Differential Buffer

Architectures Supported:

- ECP5
- LFMNX
- LIFMD
- LIFMDF

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, E

OUTPUT: Q

Description

The INRDB primitive is used to support post-PAR simulation of Dynamic Bank Controller. The Dynamic Bank Controller signals to the IO are hardwired and cannot be changed for the simulation, so the INRDB and LVDSOB primitives are defined to support the simulation.

For more information, refer to the following technical note on the Lattice web site:

- TN1198 - Power Estimation and Management for MachXO2 Device

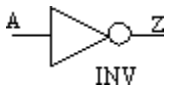
INV

Inverter

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF

- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: A

OUTPUT: Z

Note

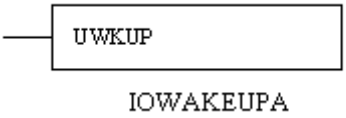
- It is possible that this primitive will be optimized away.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

IOWAKEUPA

XP2 Wake-up Controller

Architectures Supported:

- LatticeXP2



INPUT: UWKUP

Description

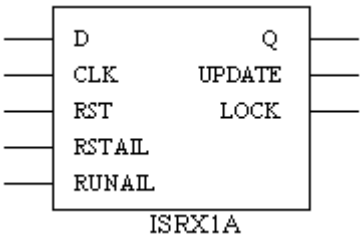
LatticeXP2 Wake-up controller.

ISRX1A

Input 1-Bit Shift Register

Architectures Supported:

- LatticeSC/M

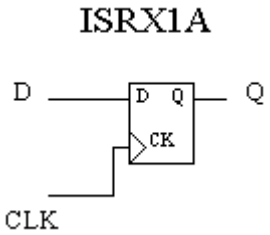


INPUTS: D, CLK, RST, RSTAIL, RUNAIL

OUTPUTS: Q, UPDATE, LOCK

Description

Shift register input logic that uses adaptive FF to capture input data. The following symbolic diagram shows the flip-flop structure of this primitive.

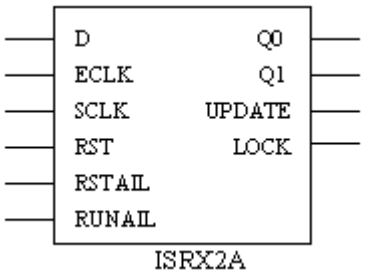


ISRX2A

Input 2-Bit Shift Register

Architectures Supported:

- LatticeSC/M

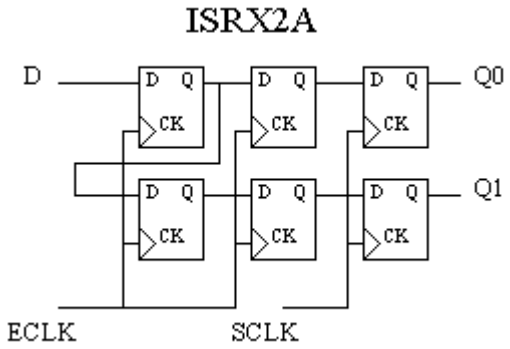


INPUTS: D, ECLK, SCLK, RST, RSTAIL, RUNAIL

OUTPUTS: Q0, Q1, UPDATE, LOCK

Description

Shift register input logic that allows clock domain transfer from edge clock to primary clock and parallel transfer to the core of incoming serial data. The following symbolic diagram shows the flip-flop structure of this primitive.

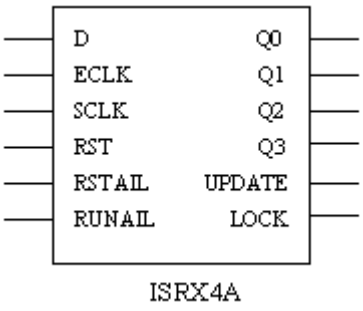


ISRX4A

Input 4-Bit Shift Register

Architectures Supported:

- LatticeSC/M

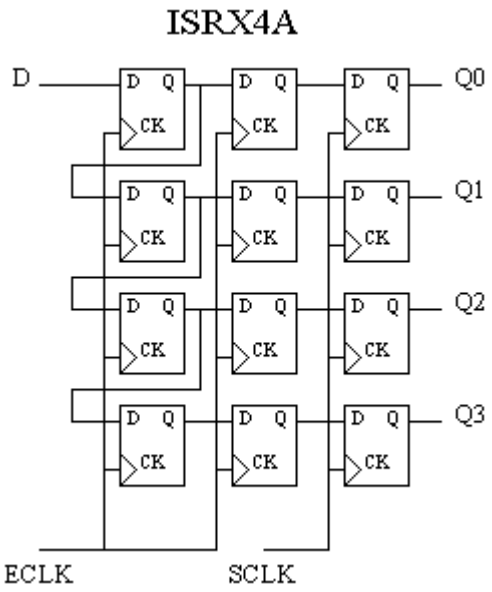


INPUTS: D, ECLK, SCLK, RST, RSTAIL, RUNAIL

OUTPUTS: Q0, Q1, Q2, Q3, UPDATE, LOCK

Description

Shift register input logic that allows clock domain transfer from edge clock to primary clock and parallel transfer to the core of incoming serial data. The following symbolic diagram shows the flip-flop structure of this primitive.



I2CA

Architectures Supported:

- LIFMD
- LIFMDF

InputL CSI;CLKI;STBI; WEI; ADRI3, ADRI2, ADRI1, ADRI0; DATI9, DATI8, DATI7, DATI6, DATI5, DATI4, DATI3, DATI2, DATI1, DATI0; DATO9, DATO8, DATO7, DATO6, DATO5, DATO4, DATO3, DATO2, DATO1, DATO0; FIFORST; MRDCMPL;SCLI;

Output: ACKO; SRWO; I2CIRQ; I2CWKUP, SRDWR; TXFIFOAE; TXFIFOE; TXFIFOF; RXFIFOE; RXFIFOAF;RXFIFOF; SCLO; SCLOE; SDAI; SDAO; SDAOE;

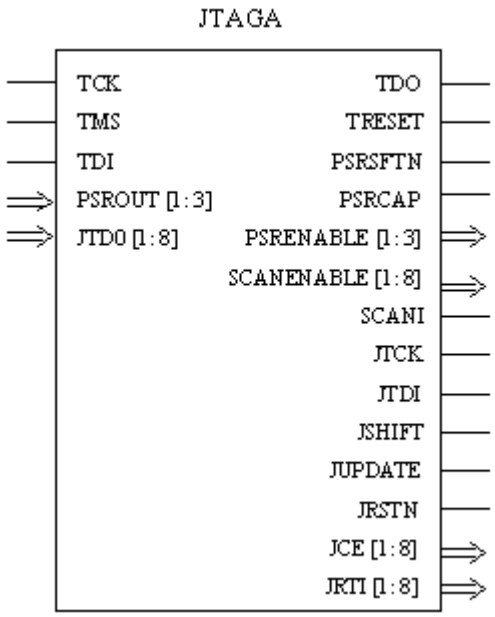
J

JTAGA

JTAG (Joint Test Action Group) Controller

Architectures Supported:

- LatticeSC/M



INPUTS: TCK, TMS, TDI, PSROUT1, PSROUT2, PSROUT3, JTDO1, JTDO2, JTDO3, JTDO4, JTDO5, JTDO6, JTDO7, JTDO8

OUTPUTS: TDO, TRESET, PSRSFTN, PSRCAP, PSRENABLE1, PSRENABLE2, PSRENABLE3, SCANENABLE1, SCANENABLE2, SCANENABLE3, SCANENABLE4, SCANENABLE5, SCANENABLE6, SCANENABLE7, SCANENABLE8, SCANI, JTCK, JTDI, JSHIFT, JUPDATE, JRSTN, JCE1, JCE2, JCE3, JCE4, JCE5, JCE6, JCE7, JCE8, JRTI1, JRTI2, JRTI3, JRTI4, JRTI5, JRTI6, JRTI7, JRTI8

Description

The JTAGA primitive provides the control and interconnect circuit used by the boundary scan function. This function allows the testing of increasingly complex ICs and IC packages. The LatticeSC/M device has enhanced its interface capability to the PLC array with increased scan chain connectivity and tap state machine flags such as shift capture update, reset, run test idle.

Example pin functions:

Table 745:

Pins	I/O	Function	Description
TCK TMS TDI TDO	I I I O	interface pins	Test clock (TCK), Test mode select (TMS), Test data in (TDI) and Test data out (TDO) are four interface pins for this primitive. The TDI, TMS and TCK pins are connected to the dedicated IO pads on the device.
PSROUT[1:3] JTDO[1:8]	I I	user boundary scan-ring outputs	These are the outputs of the last registers of the user scan rings to the boundary scan block. Inputs to the boundary scan macro are based on the instruction loaded.
TRESET	O	reset	Active high output of the boundary scan macro to the routing. The output is high when the boundary scan macro is in test logic reset state.
PSRSFTN	O	shift_not data register	Active low output of the boundary scan macro. The output is low when the boundary scan macro is in the shift data state and the programmable scan ring instructions are loaded.
PSRCAP	O	capture data register	Active high output of the boundary scan macro. The output is high when the boundary scan macro is in the capture data state and the programmable scan ring instructions are loaded.
PSRENABLE[1:3] SCANENABLE[1:8]	O	enable flag	Active high outputs of the boundary scan macro to the routing. The output equals a high upon update of the specific instructions, PLC_SCAN_RING[1:3] and SCAN[1:8], respectively.
SCANI	O	scan in	Private pin used for testing of the system bus that multiplexes the TDI and SCANOUT[11:14].
JTCK	O	test clock	The boundary scan clock which is output from the boundary scan macro to the scan rings.
JTDI	O	test data in	The output of the boundary scan macro from where the test data is output to the scan rings.
JSHIFT	O	shift data register	Active high output of the boundary scan macro. The output is high when the boundary scan macro is in the shift data state and the scan instructions are loaded.
JUPDATE	O	update data register	Active high output of the boundary scan macro. The output is high when the boundary scan macro is in the update data state and when the PLC_SCAN_RING or SCAN instructions are loaded.

Table 745:

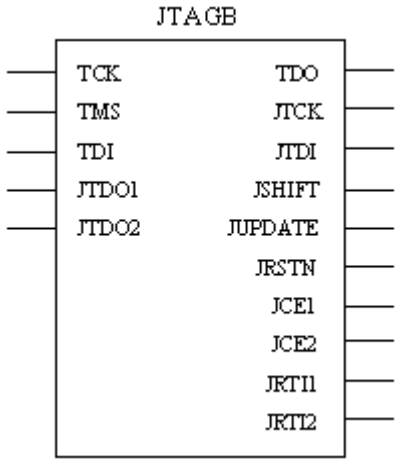
Pins	I/O	Function	Description
JRSTN	O	reset_not	Active low output of the boundary scan macro to the routing. The output is low when the boundary scan macro is in test logic reset state.
JCE[1:8]	O	clock enable	Active high output of the boundary scan macro to the routing. The output is high when the boundary scan macro is in the SHIFT or CAPTURE state during the SCAN[1:8] instructions, respectively.
JRTI[1:8]	O	run test idle	Active high output of the boundary scan macro. The output is high when the boundary scan macro is in the run test idle state and during the SCAN[1:8] instructions.

JTAGB

JTAG (Joint Test Action Group) Controller

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: TCK, TMS, TDI, JTDO1, JTDO2

OUTPUTS: TDO, JTCK, JTDI, JSHIFT, JUPDATE, JRSTN, JCE1, JCE2, JRTI1, JRTI2

ATTRIBUTES:

ER1: "ENABLED" (default), "DISABLED"

ER2: "ENABLED" (default), "DISABLED"

Description

Table 746:

Signal	I/O	Description
TCK	I	Test Clock, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TCK pin. Clocks registers and TAP Controller.
TMS	I	Test Mode Select, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TMS pin. Controls state machine switching for TAP Controller.
TDI	I	Test Data Input, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TDI pin.
JTDO[2:1]	I	JTAG Test Data Output one (scans output bus entering JTAG block), for internal logic to control non-disruptive re-configuration through JTAG port, JTAG serial interface to the device. Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO1. If ER2 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO2.
TDO	O	Test Data Output, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly to the pad of device TDO pin
JTCK	O	JTAG Test Clock (connects to TCK), for internal logic to control non-disruptive re-configuration through JTAG port. Signal is coming from TCK input and going into the FPGA fabric.
JTDI	O	JTAG Test Data Input, for internal logic to control non-disruptive re-configuration through JTAG port. Signal is coming from TDI input and going into the FPGA fabric.
JSHIFT	O	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JUPDATE	O	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JRSTN	O	JTAG Reset (active low). Signal goes low when TAP controller state is Test-Logic-Reset.

Table 746:

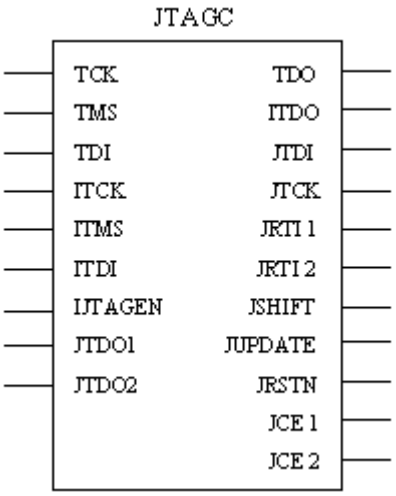
Signal	I/O	Description
JCE[2:1]	O	JTAG Clock Enable one (BS is to boundary scan ring (L2). Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JCE1 will go high when TAP controller is in Capture-DR or Shift-DR states. If ER2 instruction is shifted into the JTAG instruction register, JCE2 will go high when TAP controller is in Capture-DR or Shift-DR states.
JRTI[2:1]	O	JTAG Run-Test/Idle. Lattice supports two private JTAG instructions ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JRTI1 will go high when TAP controller is in Run-Test/Idle state. If ER2 instruction is shifted into the JTAG instruction register, JRTI2 will go high when TAP controller is in Run-Test/Idle state.

JTAGC

JTAG (Joint Test Action Group) Controller

Architectures Supported:

- LatticeECP2/M



INPUTS: TCK, TMS, TDI, ITCK, ITMS, ITDI, IJTAGEN, JTDO1, JTDO2

OUTPUTS: TDO, ITDO, JTDI, JTCK, JRTI1, JRTI2, JSHIFT, JUPDATE, JRSTN, JCE1, JCE2

Note

- The internal JTAG mode is not supported. The ITCK, ITMS, ITDI, and ITDO ports are non-operations and hence do not use them.
- The IJTAGEN port should always be tied to VCC.

ATTRIBUTES:

ER1: "ENABLED" (default), "DISABLED"

ER2: "ENABLED" (default), "DISABLED"

Description

Table 747:

Signal	I/O	Description
TCK	I	Test Clock, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TCK pin. Clocks registers and TAP Controller.
TMS	I	Test Mode Select, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TMS pin. Controls state machine switching for TAP Controller.
TDI	I	Test Data Input, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TDI pin.
ITCK	I	Internal Test Clock for internal logic to control non-disruptive re-configuration through JTAG port, comes from configuration block CIB.
ITMS	I	Internal Test Mode Select, for internal logic to control non-disruptive re-configuration through JTAG port, comes from configuration block CIB.
ITDI	I	Internal Test Data Input, for internal logic to control non-disruptive re-configuration through JTAG port, comes from configuration block CIB.
IJTAGEN	I	Internal JTAG Enable (active low), for internal logic to control non-disruptive re-configuration through JTAG port, comes from configuration block CIB.
JTDO[2:1]	I	JTAG Test Data Output one (scans output bus entering JTAG block), for internal logic to control non-disruptive re-configuration through JTAG port, JTAG serial interface to the device. Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO1. If ER2 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO2.
TDO	O	Test Data Output, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly to the pad of device TDO pin.

Table 747:

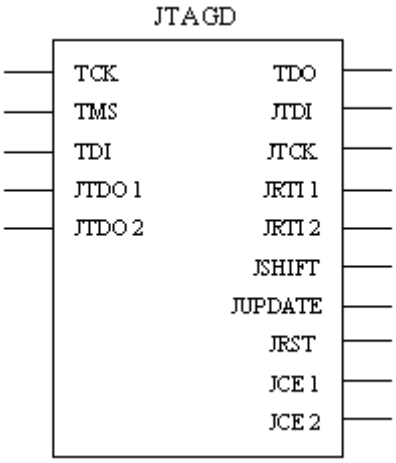
Signal	I/O	Description
ITDO	O	Internal Test Data Output for internal logic to control non-disruptive re-configuration through JTAG port, comes from configuration block CIB.
JTDI	O	JTAG Test Data Input, for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TDI input and going into the FPGA fabric.
JTCK	O	JTAG Test Clock (connects to TCK), for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TCK input and going into the FPGA fabric.
JRTI[2:1]	O	JTAG Run-Test/Idle. Lattice supports two private JTAG instructions ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JRTI1 will go high when TAP controller is in Run-Test/Idle state. If ER2 instruction is shifted into the JTAG instruction register, JRTI2 will go high when TAP controller is in Run-Test/Idle state.
JSHIFT	O	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JUPDATE	O	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JRSTN	O	JTAG Reset (active low). Signal goes low when TAP controller state is Test-Logic-Reset.
JCE[2:1]	O	JTAG Clock Enable one (BS is to boundary scan ring (L2). Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JCE1 will go high when TAP controller is in Capture-DR or Shift-DR states. If ER2 instruction is shifted into the JTAG instruction register, JCE2 will go high when TAP controller is in Capture-DR or Shift-DR states.

JTAGD

JTAG (Joint Test Action Group) Controller

Architectures Supported:

- MachXO
- Platform Manager



INPUTS: TCK, TMS, TDI, JTDO1, JTDO2

OUTPUTS: TDO, JTDI, JTCK, JRTI1, JRTI2, JSHIFT, JUPDATE, JRST, JCE1, JCE2

ATTRIBUTES:

ER1: "ENABLED" (default), "DISABLED"

ER2: "ENABLED" (default), "DISABLED"

Description

Table 748:

Signal	I/O	Description
TCK	I	Test Clock, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TCK pin Clocks registers and TAP Controller
TMS	I	Test Mode Select, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TMS pin Controls state machine switching for TAP Controller
TDI	I	Test Data Input, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TDI pin
JTDO[2:1]	I	JTAG Test Data Output one (scans output bus entering JTAG block), for internal logic to control non-disruptive re-configuration through JTAG port, JTAG serial interface to the device. Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO1. If ER2 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO2.

Table 748:

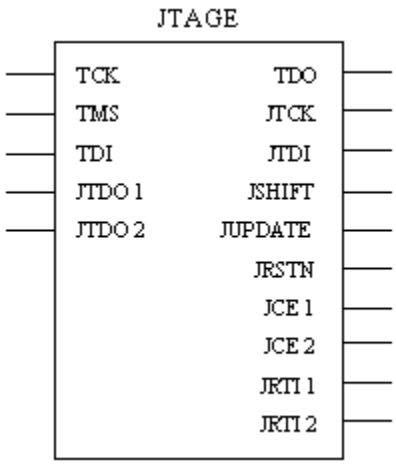
Signal	I/O	Description
TDO	O	Test Data Output, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly to the pad of device TDO pin
JTDI	O	JTAG Test Data Input, for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TDI input and going into the FPGA fabric.
JTCK	O	JTAG Test Clock (connects to TCK), for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TCK input and going into the FPGA fabric.
JRTI[2:1]	O	JTAG Run-Test/Idle Lattice supports two private JTAG instructions ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JRTI1 will go high when TAP controller is in Run-Test/Idle state. If ER2 instruction is shifted into the JTAG instruction register, JRTI2 will go high when TAP controller is in Run-Test/Idle state.
JSHIFT	O	JTAG Shift Signal goes high when TAP Controller State is Shift-DR.
JUPDATE	O	JTAG Update Signal goes high when TAP controller state is Update-DR.
JRST	O	JTAG Reset (active high) Signal goes high when TAP controller state is Test-Logic-Reset.
JCE[2:1]	O	JTAG Clock Enable one (BS is to boundary scan ring (L2). Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JCE1 will go high when TAP controller is in Capture-DR or Shift-DR states. If ER2 instruction is shifted into the JTAG instruction register, JCE2 will go high when TAP controller is in Capture-DR or Shift-DR states.

JTAGE

JTAG (Joint Test Action Group) Controller

Architectures Supported:

- LatticeECP3
- LatticeXP2



INPUTS: TCK, TMS, TDI, JTDO1, JTDO2

OUTPUTS: TDO, JTCK, JTDI, JSHIFT, JUPDATE, JRSTN, JCE1, JCE2, JRTI1, JRTI2

ATTRIBUTES:

ER1: "ENABLED" (default), "DISABLED"

ER2: "ENABLED" (default), "DISABLED"

Description

Table 749:

Signal	I/O	Description
TCK	I	Test Clock, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TCK pin. Clocks registers and TAP Controller.
TMS	I	Test Mode Select, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TMS pin. Controls state machine switching for TAP Controller.
TDI	I	Test Data Input, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TDI pin.
JTDO[2:1]	I	JTAG Test Data Output one (scans output bus entering JTAG block), for internal logic to control non-disruptive re-configuration through JTAG port, JTAG serial interface to the device. Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO1. If ER2 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO2.

Table 749:

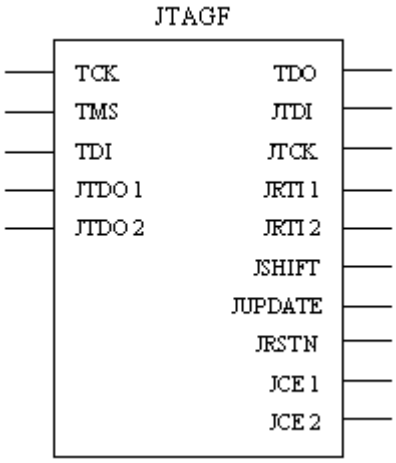
Signal	I/O	Description
TDO	O	Test Data Output, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly to the pad of device TDO pin.
JTDI	O	JTAG Test Data Input, for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TDI input and going into the FPGA fabric.
JTCK	O	JTAG Test Clock (connects to TCK), for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TCK input and going into the FPGA fabric.
JRTI[2:1]	O	JTAG Run-Test/Idle. Lattice supports two private JTAG instructions ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JRTI1 will go high when TAP controller is in Run-Test/Idle state. If ER2 instruction is shifted into the JTAG instruction register, JRTI2 will go high when TAP controller is in Run-Test/Idle state.
JSHIFT	O	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JRSTN	O	JTAG Reset (active low) Signal goes low when TAP controller state is Test-Logic-Reset.
JUPDATE	O	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JCE[2:1]	O	JTAG Clock Enable one (BS is to boundary scan ring (L2). Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JCE1 will go high when TAP controller is in Capture-DR or Shift-DR states. If ER2 instruction is shifted into the JTAG instruction register, JCE2 will go high when TAP controller is in Capture-DR or Shift-DR states.

JTAGF

JTAG (Joint Test Action Group) Controller

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: TCK, TMS, TDI, JTDO1, JTDO2

OUTPUTS: TDO, JTDI, JTCK, JRTI1, JRTI2, JSHIFT, JUPDATE, JRSTN, JCE1, JCE2

ATTRIBUTES:

ER1: "ENABLED" (default), "DISABLED"

ER2: "ENABLED" (default), "DISABLED"

Description

The JTAGF primitive is used to provide access to internal JTAG signals from within the FPGA fabric. This is used for some cores, such as REVEAL, and other purposes. It is not a component an ordinary user would normally use directly.

Table 750:

Signal	I/O	Description
TCK	I	Test Clock, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TCK pin. Clocks registers and TAP Controller.
TMS	I	Test Mode Select, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TMS pin. Controls state machine switching for TAP Controller.
TDI	I	Test Data Input, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly from the pad of device TDI pin.

Table 750:

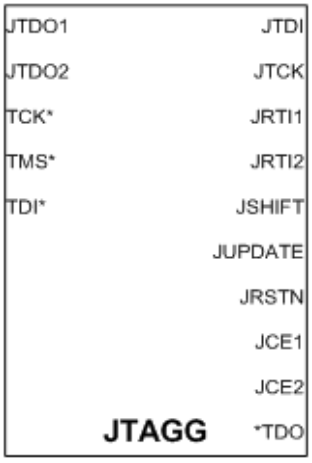
Signal	I/O	Description
JTDO[2:1]	I	JTAG Test Data Output one (scans output bus entering JTAG block), for internal logic to control non-disruptive re-configuration through JTAG port, JTAG serial interface to the device. Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO1. If ER2 instruction is shifted into the JTAG instruction register, TDO output will come from JTDO2.
TDO	O	Test Data Output, one of the pins for the Test Access Port (TAP), JTAG serial interface to the device, connected directly to the pad of device TDO pin.
JTDI	O	JTAG Test Data Input, for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TDI input and going into the FPGA fabric.
JTCK	O	JTAG Test Clock (connects to TCK), for internal logic to control non-disruptive re-configuration through JTAG port Signal is coming from TCK input and going into the FPGA fabric.
JRTI[2:1]	O	JTAG Run-Test/Idle. Lattice supports two private JTAG instructions ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JRTI1 will go high when TAP controller is in Run-Test/Idle state. If ER2 instruction is shifted into the JTAG instruction register, JRTI2 will go high when TAP controller is in Run-Test/Idle state.
JSHIFT	O	JTAG Shift. Signal goes high when TAP Controller State is Shift-DR.
JRSTN	O	JTAG Reset (active low) Signal goes low when TAP controller state is Test-Logic-Reset.
JUPDATE	O	JTAG Update. Signal goes high when TAP controller state is Update-DR.
JCE[2:1]	O	JTAG Clock Enable one (BS is to boundary scan ring (L2). Lattice supports two private JTAG instructions, ER1 (0x32) and ER2 (0x38). If ER1 instruction is shifted into the JTAG instruction register, JCE1 will go high when TAP controller is in Capture-DR or Shift-DR states. If ER2 instruction is shifted into the JTAG instruction register, JCE2 will go high when TAP controller is in Capture-DR or Shift-DR states.

JTAGG

JTAG (Joint Test Action Group) Controller

Architectures Supported:

- ECP5



INPUTS: TCK, TMS, TDI, JTDO2, JTDO1

OUTPUTS: TDO, JTDI, JTCK, JRTI2, JRTI1, JSHIFT, JUPDATE, JRSTN, JCE2, JCE1

Description

The JTAGG element is used to provide access to internal JTAG signals from within the FPGA fabric. This element is used for some cores, such as Reveal Logic Analyzer, and other purposes. Most users would typically not use this component directly.

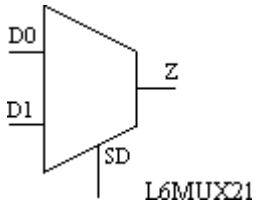
L

L6MUX21

2 to 1 Mux

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SD

OUTPUT: Z

Truth Table

Table 751:

INPUTS			OUTPUTS
D0	D1	SD	Z
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1

Note

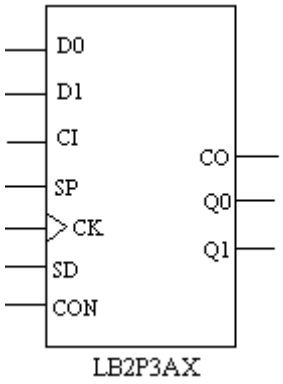
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LB2P3AX

2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CON

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 752:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	CON	CO	Q[0:1]
D[0:1]	1	0	1	↑	1	0	D[0:1]
D[0:1]	1	1	1	↑	1	*	D[0:1]
X	0	0	X	X	1	0	Q[0:1]
X	X	0	0	X	1	0	Q[0:1]
X	X	1	0	X	1	*	Q[0:1]
X	0	1	1	↑	1	*	count+1
D[0:1]	1	1	1	↑	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	**	D[0:1]
X	0	1	X	X	0	1	Q[0:1]
X	X	1	0	X	0	1	Q[0:1]
X	X	0	0	X	0	**	Q[0:1]
X	0	0	1	↑	0	**	count-1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

** When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=!CON•CI, Q[0:1]=0 (D[0:1]=SP=CK=SD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

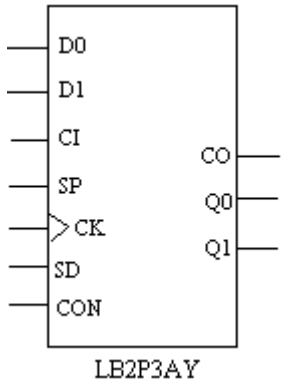
LB2P3AY

2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M

- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CON

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 753:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	CON	CO	Q[0:1]
D[0:1]	1	0	1	↑	1	0	D[0:1]
D[0:1]	1	1	1	↑	1	*	D[0:1]
X	0	0	X	X	1	0	Q[0:1]
X	X	0	0	X	1	0	Q[0:1]
X	X	1	0	X	1	*	Q[0:1]
X	0	1	1	↑	1	*	count+1
D[0:1]	1	1	1	↑	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	**	D[0:1]
X	0	1	X	X	0	1	Q[0:1]
X	X	1	0	X	0	1	Q[0:1]
X	X	0	0	X	0	**	Q[0:1]
X	0	0	1	↑	0	**	count-1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

** When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, Q[0:1]=1, CO=!CON+CI (D[0:1]=SP=CK=SD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

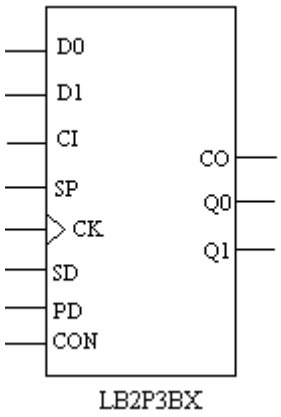
LB2P3BX

2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M

- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, PD, CON

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 754:

INPUTS							OUTPUTS	
D[0:1]	SD	CI	SP	CK	CON	PD	CO	Q[0:1]
X	X	0	X	X	1	1	0	1
X	X	1	X	X	1	1	1	1
D[0:1]	1	0	1	↑	1	0	0	D[0:1]
D[0:1]	1	1	1	↑	1	0	*	D[0:1]
X	0	0	X	X	1	0	0	Q[0:1]
X	X	0	0	X	1	0	0	Q[0:1]
X	X	1	0	X	1	0	*	Q[0:1]
X	0	1	1	↑	1	0	*	count+1
X	X	X	X	X	0	1	1	1
D[0:1]	1	1	1	↑	0	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	0	**	D[0:1]
X	0	1	X	X	0	0	1	Q[0:1]
X	X	1	0	X	0	0	1	Q[0:1]
X	X	0	0	X	0	0	**	Q[0:1]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

** When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=!CON+CI, Q[0:1]=1 (D[0:1]=SP=CK=SD=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

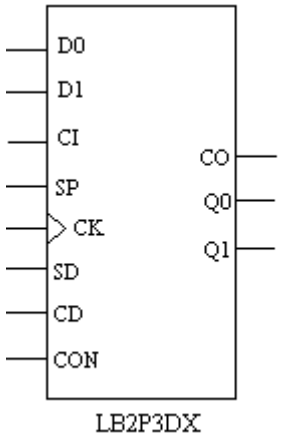
LB2P3DX

2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC

- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CD, CON

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 755:

INPUTS							OUTPUTS	
D[0:1]	SD	CI	SP	CK	CON	CD	CO	Q[0:1]
X	X	X	X	X	1	1	0	0
D[0:1]	1	0	1	↑	1	0	0	D[0:1]
D[0:1]	1	1	1	↑	1	0	*	D[0:1]
X	0	0	X	X	1	0	0	Q[0:1]
X	X	0	0	X	1	0	0	Q[0:1]
X	X	1	0	X	1	0	*	Q[0:1]
X	0	1	1	↑	1	0	*	count+1
X	X	0	X	X	0	1	0	0
X	X	1	X	X	0	1	1	0
D[0:1]	1	1	1	↑	0	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	0	**	D[0:1]
X	0	1	X	X	0	0	1	Q[0:1]
X	X	1	0	X	0	0	1	Q[0:1]
X	X	0	0	X	0	0	**	Q[0:1]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

** When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=!CON"CI, Q[0:1]=0 (D[0:1]=SP=CK=SD=CD=X)

Note

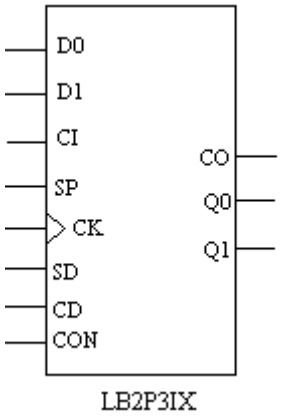
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LB2P3IX

2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CD, CON

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 756:

INPUTS							OUTPUTS	
D[0:1]	SD	CI	SP	CK	CON	CD	CO	Q[0:1]
X	X	X	X	↑	1	1	0	0
D[0:1]	1	0	1	↑	1	0	0	D[0:1]
D[0:1]	1	1	1	↑	1	0	*	D[0:1]
X	0	0	X	X	1	0	0	Q[0:1]
X	X	0	0	X	1	0	0	Q[0:1]
X	X	1	0	X	1	0	*	Q[0:1]
X	0	1	1	↑	1	0	*	count+1
X	X	0	X	↑	0	1	0	0
X	X	1	X	↑	0	1	1	0
D[0:1]	1	1	1	↑	0	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	0	**	D[0:1]
X	0	1	X	X	0	0	1	Q[0:1]
X	X	1	0	X	0	0	1	Q[0:1]
X	X	0	0	X	0	0	**	Q[0:1]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

** When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=!CON"CI, Q[0:1]=0 (D[0:1]=SP=CK=SD=CD=X)

Note

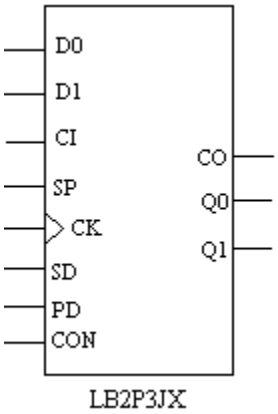
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LB2P3JX

2 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, PD, CON

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 757:

INPUTS							OUTPUTS	
D[0:1]	SD	CI	SP	CK	CON	PD	CO	Q[0:1]
X	X	0	X	↑	1	1	0	1
X	X	1	X	↑	1	1	1	1
D[0:1]	1	0	1	↑	1	0	0	D[0:1]
D[0:1]	1	1	1	↑	1	0	*	D[0:1]
X	0	0	X	X	1	0	0	Q[0:1]
X	X	0	0	X	1	0	0	Q[0:1]
X	X	1	0	X	1	0	*	Q[0:1]
X	0	1	1	↑	1	0	*	count+1
X	X	X	X	↑	0	1	1	1
D[0:1]	1	1	1	↑	0	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	0	**	D[0:1]
X	0	1	X	X	0	0	1	Q[0:1]
X	X	1	0	X	0	0	1	Q[0:1]
X	X	0	0	X	0	0	**	Q[0:1]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

** When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=!CON+CI, Q[0:1]=1 (D[0:1]=SP=CK=SD=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

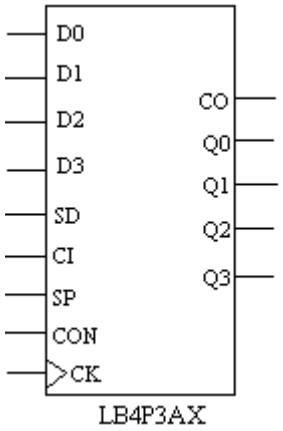
LB4P3AX

4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Clear

Architectures Supported:

- LatticeECP/EC

- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CON

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 758:

INPUTS					OUTPUTS		
D[0:3]	SD	CI	SP	CK	CON	CO	Q[0:3]
D[0:3]	1	0	1	↑	1	0	D[0:3]
D[0:3]	1	1	1	↑	1	*	D[0:3]
X	0	0	X	X	1	0	Q[0:3]
X	X	0	0	X	1	0	Q[0:3]
X	X	1	0	X	1	*	Q[0:3]
X	0	1	1	↑	1	*	count+1
D[0:3]	1	1	1	↑	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	**	D[0:3]
X	0	1	X	X	0	1	Q[0:3]
X	X	1	0	X	0	1	Q[0:3]
X	X	0	0	X	0	**	Q[0:3]
X	0	0	1	↑	0	**	count-1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

** When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

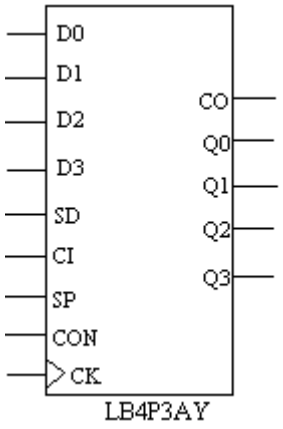
When GSR=0, CO=!CON•CI, Q[0:3]=0 (D[0:3]=SP=CK=SD=X)

LB4P3AY

4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable, GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CON

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 759:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	CON	CO	Q[0:3]
D[0:3]	1	0	1	↑	1	0	D[0:3]
D[0:3]	1	1	1	↑	1	*	D[0:3]
X	0	0	X	X	1	0	Q[0:3]
X	X	0	0	X	1	0	Q[0:3]
X	X	1	0	X	1	*	Q[0:3]
X	0	1	1	↑	1	*	count+1
D[0:3]	1	1	1	↑	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	**	D[0:3]
X	0	1	X	X	0	1	Q[0:3]
X	X	1	0	X	0	1	Q[0:3]
X	X	0	0	X	0	**	Q[0:3]
X	0	0	1	↑	0	**	count-1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

** When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

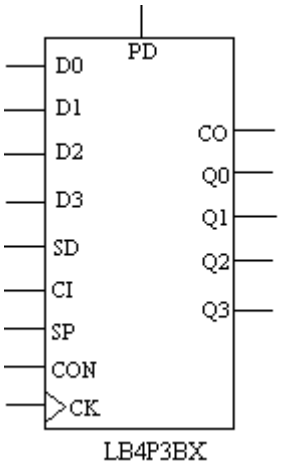
When GSR=0, Q[0:3]=1, CO=!CON+CI (D[0:3]=SP=CK=SD=X)

LB4P3BX

4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, PD, CON

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 760:

INPUTS							OUTPUTS	
D[0:3]	SD	CI	SP	CK	CON	PD	CO	Q[0:3]
X	X	0	X	X	1	1	0	1
X	X	1	X	X	1	1	1	1
D[0:3]	1	0	1	↑	1	0	0	D[0:3]
D[0:3]	1	1	1	↑	1	0	*	D[0:3]
X	0	0	X	X	1	0	0	Q[0:3]
X	X	0	0	X	1	0	0	Q[0:3]
X	X	1	0	X	1	0	*	Q[0:3]
X	0	1	1	↑	1	0	*	count+1
X	X	X	X	X	0	1	1	1
D[0:3]	1	1	1	↑	0	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	0	**	D[0:3]
X	0	1	X	X	0	0	1	Q[0:3]
X	X	1	0	X	0	0	1	Q[0:3]
X	X	0	0	X	0	0	**	Q[0:3]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

** When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

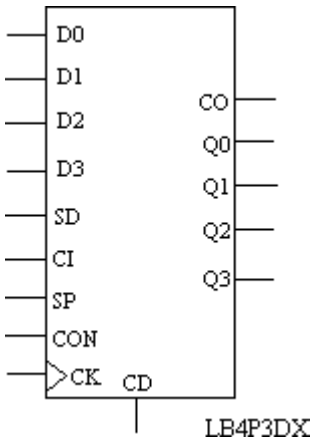
When GSR=0, CO=!CON+CI, Q[0:3]=1 (D[0:3]=SP=CK=SD=PD=X)

LB4P3DX

4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CD, CON

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 761:

INPUTS							OUTPUTS	
D[0:3]	SD	CI	SP	CK	CON	CD	CO	Q[0:3]
X	X	X	X	X	1	1	0	0
D[0:3]	1	0	1	↑	1	0	0	D[0:3]
D[0:3]	1	1	1	↑	1	0	*	D[0:3]
X	0	0	X	X	1	0	0	Q[0:3]
X	X	0	0	X	1	0	0	Q[0:3]
X	X	1	0	X	1	0	*	Q[0:3]
X	0	1	1	↑	1	0	*	count+1
X	X	0	X	X	0	1	0	0
X	X	1	X	X	0	1	1	0
D[0:3]	1	1	1	↑	0	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	0	**	D[0:3]
X	0	1	X	X	0	0	1	Q[0:3]
X	X	1	0	X	0	0	1	Q[0:3]
X	X	0	0	X	0	0	**	Q[0:3]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

** When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

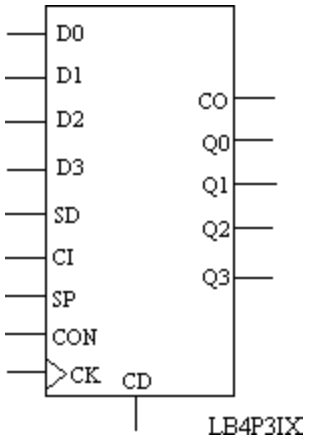
When GSR=0, CO=!CON"CI, Q[0:3]=0 (D[0:3]=SP=CK=SD=CD=X)

LB4P3IX

4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CD, CON

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 762:

INPUTS							OUTPUTS	
D[0:3]	SD	CI	SP	CK	CON	CD	CO	Q[0:3]
X	X	X	X	↑	1	1	0	0
D[0:3]	1	0	1	↑	1	0	0	D[0:3]
D[0:3]	1	1	1	↑	1	0	*	D[0:3]
X	0	0	X	X	1	0	0	Q[0:3]
X	X	0	0	X	1	0	0	Q[0:3]
X	X	1	0	X	1	0	*	Q[0:3]
X	0	1	1	↑	1	0	*	count+1
X	X	0	X	↑	0	1	0	0
X	X	1	X	↑	0	1	1	0
D[0:3]	1	1	1	↑	0	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	0	**	D[0:3]
X	0	1	X	X	0	0	1	Q[0:3]
X	X	1	0	X	0	0	1	Q[0:3]
X	X	0	0	X	0	0	**	Q[0:3]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

** When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

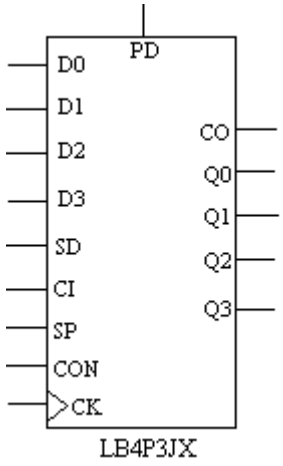
When GSR=0, CO=!CON"CI, Q[0:3]=0 (D[0:3]=SP=CK=SD=CD=X)

LB4P3JX

4 Bit Positive Edge Triggered Loadable Bidirectional Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, PD, CON

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

When CON = 0, CI and CO are active LOW

Truth Table

Table 763:

INPUTS							OUTPUTS	
D[0:3]	SD	CI	SP	CK	CON	PD	CO	Q[0:3]
X	X	0	X	↑	1	1	0	1
X	X	1	X	↑	1	1	1	1
D[0:3]	1	0	1	↑	1	0	0	D[0:3]
D[0:3]	1	1	1	↑	1	0	*	D[0:3]
X	0	0	X	X	1	0	0	Q[0:3]
X	X	0	0	X	1	0	0	Q[0:3]
X	X	1	0	X	1	0	*	Q[0:3]
X	0	1	1	↑	1	0	*	count+1
X	X	X	X	↑	0	1	1	1
D[0:3]	1	1	1	↑	0	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	0	**	D[0:3]
X	0	1	X	X	0	0	1	Q[0:3]
X	X	1	0	X	0	0	1	Q[0:3]
X	X	0	0	X	0	0	**	Q[0:3]
X	0	0	1	↑	0	0	**	count-1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

** When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=!CON+CI, Q[0:3]=1 (D[0:3]=SP=CK=SD=PD=X)

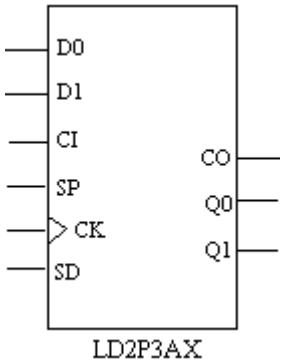
LD2P3AX

2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Clear

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP

- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

CI and CO are active LOW

Truth Table

Table 764:

INPUTS					OUTPUTS	
D[0:1]	SD	CI	SP	CK	CO	Q[0:1]
D[0:1]	1	1	1	↑	1	D[0:1]
D[0:1]	1	0	1	↑	*	D[0:1]
X	0	1	X	X	1	Q[0:1]
X	X	1	0	X	1	Q[0:1]
X	X	0	0	X	*	Q[0:1]
X	0	0	1	↑	*	count-1

X = Don't care

* When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=CI, Q[0:1]=0 (D[0:1]=SP=CK=SD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

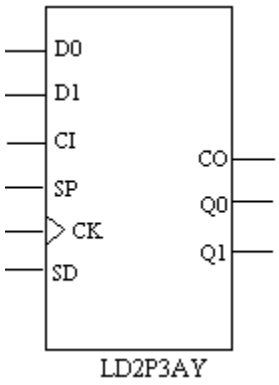
LD2P3AY

2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L

- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

CI and CO are active LOW

Truth Table

Table 765:

INPUTS					OUTPUTS	
D[0:1]	SD	CI	SP	CK	CO	Q[0:1]
D[0:1]	1	1	1	↑	1	D[0:1]
D[0:1]	1	0	1	↑	*	D[0:1]
X	0	1	X	X	1	Q[0:1]
X	X	1	0	X	1	Q[0:1]
X	X	0	0	X	*	Q[0:1]
X	0	0	1	↑	*	count-1

X = Don’t care

* When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=1, Q[0:1]=1 (D[0:1]=CI=SP=CK=SD=X)

Note

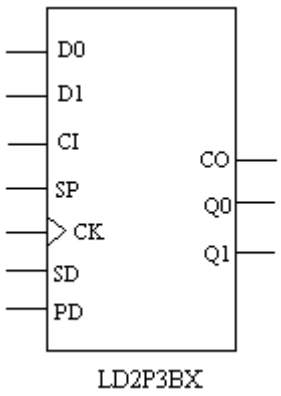
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LD2P3BX

2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

CI and CO are active LOW

Truth Table

Table 766:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	PD	CO	Q[0:1]
X	X	X	X	X	1	1	1
D[0:1]	1	1	1	↑	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	*	D[0:1]
X	0	1	X	X	0	1	Q[0:1]
X	X	1	0	X	0	1	Q[0:1]
X	X	0	0	X	0	*	Q[0:1]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=1, Q[0:1]=1 (D[0:1]=SP=CK=SD=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

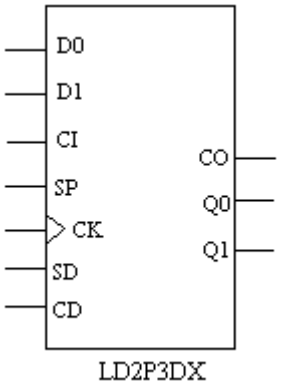
LD2P3DX

2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M

- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

CI and CO are active LOW

Truth Table

Table 767:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	CD	CO	Q[0:1]
X	X	0	X	X	1	0	0
X	X	1	X	X	1	1	0
D[0:1]	1	1	1	↑	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	*	D[0:1]
X	0	1	X	X	0	1	Q[0:1]
X	X	1	0	X	0	1	Q[0:1]
X	X	0	0	X	0	*	Q[0:1]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=CI, Q[0:1]=0 (D[0:1]=SP=CK=SD=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

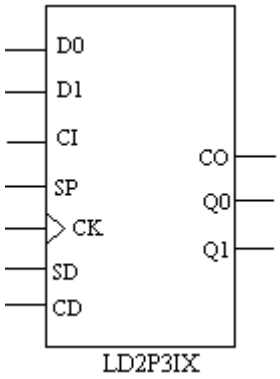
LD2P3IX

2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

CI and CO are active LOW

Truth Table

Table 768:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	CD	CO	Q[0:1]
X	X	0	X	↑	1	0	0
X	X	1	X	↑	1	1	0
D[0:1]	1	1	1	↑	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	*	D[0:1]
X	0	1	X	X	0	1	Q[0:1]
X	X	1	0	X	0	1	Q[0:1]
X	X	0	0	X	0	*	Q[0:1]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=CI, Q[0:1]=0 (D[0:1]=SP=CK=SD=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

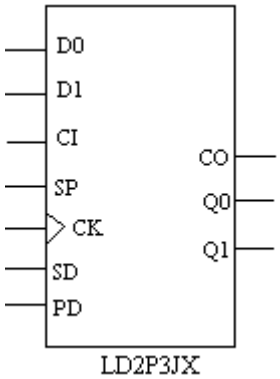
LD2P3JX

2 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

CI and CO are active LOW

Truth Table

Table 769:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	PD	CO	Q[0:1]
X	X	X	X	↑	1	1	1
D[0:1]	1	1	1	↑	0	1	D[0:1]
D[0:1]	1	0	1	↑	0	*	D[0:1]
X	0	1	X	X	0	1	Q[0:1]
X	X	1	0	X	0	1	Q[0:1]
X	X	0	0	X	0	*	Q[0:1]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:1] is 00, CO will be 0; otherwise, CO will be 1

When GSR=0, CO=1, Q[0:1]=1 (D[0:1]=SP=CK=SD=PD=X)

Note

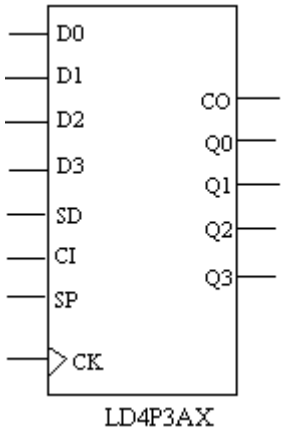
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LD4P3AX

4 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

CI and CO are active LOW

Truth Table

Table 770:

INPUTS					OUTPUTS	
D[0:3]	SD	CI	SP	CK	CO	Q[0:3]
D[0:3]	1	1	1	↑	1	D[0:3]
D[0:3]	1	0	1	↑	*	D[0:3]
X	0	1	X	X	1	Q[0:3]
X	X	1	0	X	1	Q[0:3]
X	X	0	0	X	*	Q[0:3]
X	0	0	1	↑	*	count-1

X = Don’t care

* When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

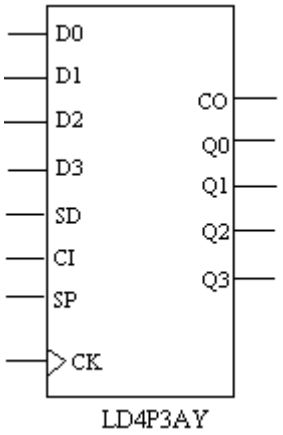
When GSR=0, CO=CI, Q[0:3]=0 (D[0:3]=SP=CK=SD=X)

LD4P3AY

4 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable, GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

CI and CO are active LOW

Truth Table

Table 771:

INPUTS					OUTPUTS	
D[0:3]	SD	CI	SP	CK	CO	Q[0:3]
D[0:3]	1	1	1	↑	1	D[0:3]
D[0:3]	1	0	1	↑	*	D[0:3]
X	0	1	X	X	1	Q[0:3]
X	X	1	0	X	1	Q[0:3]
X	X	0	0	X	*	Q[0:3]
X	0	0	1	↑	*	count-1

X = Don’t care

* When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

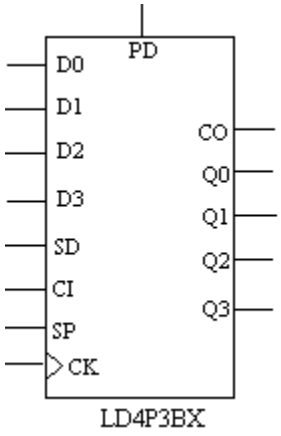
When GSR=0, CO=1, Q[0:3]=1 (D[0:3]=CI=SP=CK=SD=X)

LD4P3BX

4 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

CI and CO are active LOW

Truth Table

Table 772:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	PD	CO	Q[0:3]
X	X	X	X	X	1	1	1
D[0:3]	1	1	1	↑	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	*	D[0:3]
X	0	1	X	X	0	1	Q[0:3]
X	X	1	0	X	0	1	Q[0:3]
X	X	0	0	X	0	*	Q[0:3]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

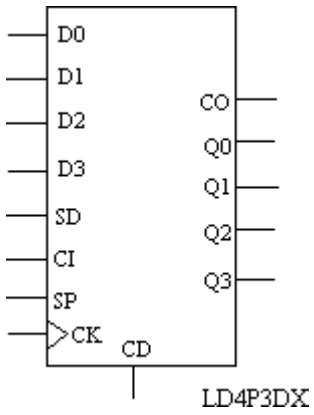
When GSR=0, CO=1, Q[0:3]=1 (D[0:3]=SP=CK=SD=PD=X)

LD4P3DX

4 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

CI and CO are active LOW

Truth Table

Table 773:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	CD	CO	Q[0:3]
X	X	0	X	X	1	0	0
X	X	1	X	X	1	1	0
D[0:3]	1	1	1	↑	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	*	D[0:3]
X	0	1	X	X	0	1	Q[0:3]
X	X	1	0	X	0	1	Q[0:3]
X	X	0	0	X	0	*	Q[0:3]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

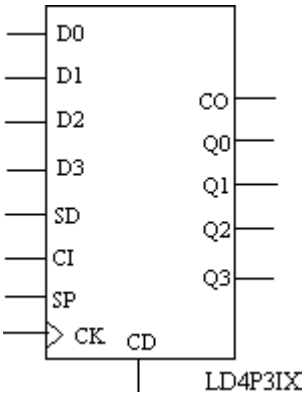
When GSR=0, CO=CI, Q[0:3]=0 (D[0:3]=SP=CK=SD=CD=X)

LD4P3IX

4 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

CI and CO are active LOW

Truth Table

Table 774:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	CD	CO	Q[0:3]
X	X	0	X	↑	1	0	0
X	X	1	X	↑	1	1	0
D[0:3]	1	1	1	↑	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	*	D[0:3]
X	0	1	X	X	0	1	Q[0:3]
X	X	1	0	X	0	1	Q[0:3]
X	X	0	0	X	0	*	Q[0:3]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

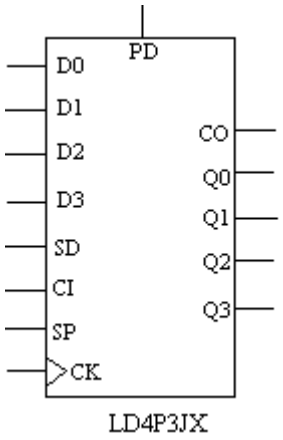
When GSR=0, CO=CI, Q[0:3]=0 (D[0:3]=SP=CK=SD=CD=X)

LD4P3JX

4 Bit Positive Edge Triggered Loadable Down-Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Note

CI and CO are active LOW

Truth Table

Table 775:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	PD	CO	Q[0:3]
X	X	X	X	↑	1	1	1
D[0:3]	1	1	1	↑	0	1	D[0:3]
D[0:3]	1	0	1	↑	0	*	D[0:3]
X	0	1	X	X	0	1	Q[0:3]
X	X	1	0	X	0	1	Q[0:3]
X	X	0	0	X	0	*	Q[0:3]
X	0	0	1	↑	0	*	count-1

X = Don’t care

* When Q[0:3] is 0000, CO will be 0; otherwise, CO will be 1

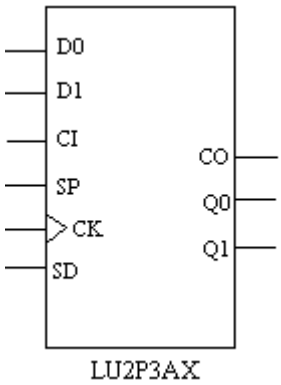
When GSR=0, CO=1, Q[0:3]=1 (D[0:3]=SP=CK=SD=PD=X)

LU2P3AX

2 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable, GSR Used for Clear

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 776:

INPUTS					OUTPUTS	
D[0:1]	SD	CI	SP	CK	CO	Q[0:1]
D[0:1]	1	0	1	↑	0	D[0:1]
D[0:1]	1	1	1	↑	*	D[0:1]
X	0	0	X	X	0	Q[0:1]
X	X	0	0	X	0	Q[0:1]
X	X	1	0	X	*	Q[0:1]
X	0	1	1	↑	*	count+1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

When GSR=0, CO=0, Q[0:1]=0 (D[0:1]=SP=CK=SD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

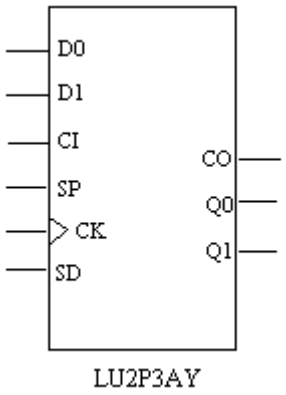
LU2P3AY

2 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable, GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L

- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 777:

INPUTS					OUTPUTS	
D[0:1]	SD	CI	SP	CK	CO	Q[0:1]
D[0:1]	1	0	1	↑	0	D[0:1]
D[0:1]	1	1	1	↑	*	D[0:1]
X	0	0	X	X	0	Q[0:1]
X	X	0	0	X	0	Q[0:1]
X	X	1	0	X	*	Q[0:1]
X	0	1	1	↑	*	count+1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

When GSR=0, CO=CI, Q[0:1]=1 (D[0:1]=CI=SP=CK=SD=X)

Note

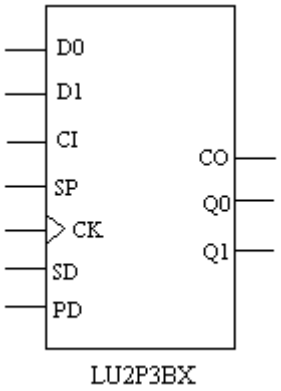
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LU2P3BX

2 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 778:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	PD	CO	Q[0:1]
X	X	0	X	X	1	0	1
X	X	1	X	X	1	1	1
D[0:1]	1	0	1	↑	0	0	D[0:1]
D[0:1]	1	1	1	↑	0	*	D[0:1]
X	0	0	X	X	0	0	Q[0:1]
X	X	0	0	X	0	0	Q[0:1]
X	X	1	0	X	0	*	Q[0:1]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

When GSR=0, CO=CI, Q[0:1]=1 (D[0:1]=SP=CK=SD=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

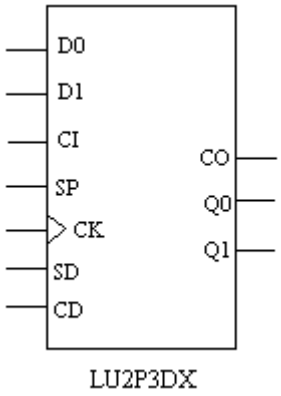
LU2P3DX

2 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 779:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	CD	CO	Q[0:1]
X	X	X	X	X	1	0	0
D[0:1]	1	0	1	↑	0	0	D[0:1]
D[0:1]	1	1	1	↑	0	*	D[0:1]
X	0	0	X	X	0	0	Q[0:1]
X	X	0	0	X	0	0	Q[0:1]
X	X	1	0	X	0	*	Q[0:1]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

When GSR=0, CO=0, Q[0:1]=0 (D[0:1]=SP=CK=SD=CD=X)

Note

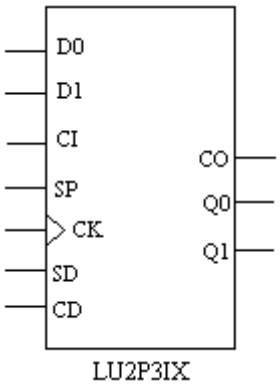
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LU2P3IX

2 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 780:

INPUTS						OUTPUTS	
D[0:1]	SD	CI	SP	CK	CD	CO	Q[0:1]
X	X	X	X	↑	1	0	0
D[0:1]	1	0	1	↑	0	0	D[0:1]
D[0:1]	1	1	1	↑	0	*	D[0:1]
X	0	0	X	X	0	0	Q[0:1]
X	X	0	0	X	0	0	Q[0:1]
X	X	1	0	X	0	*	Q[0:1]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

When GSR=0, CO=0, Q[0:1]=0 (D[0:1]=SP=CK=SD=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

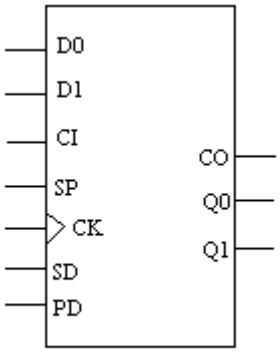
LU2P3JX

2 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Truth Table

Table 781:

INPUTS						**OUTPUTS**	
D[0:1]	SD	CI	SP	CK	PD	CO	Q[0:1]
X	X	0	X	↑	1	0	1
X	X	1	X	↑	1	1	1
D[0:1]	1	0	1	↑	0	0	D[0:1]
D[0:1]	1	1	1	↑	0	*	D[0:1]
X	0	0	X	X	0	0	Q[0:1]
X	X	0	0	X	0	0	Q[0:1]
X	X	1	0	X	0	*	Q[0:1]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:1] is 11, CO will be 1; otherwise, CO will be 0

When GSR=0, CO=CI, Q[0:1]=1 (D[0:1]=SP=CK=SD=PD=X)

Note

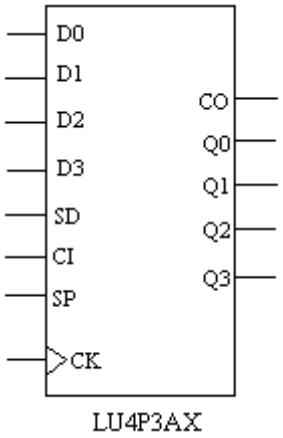
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

LU4P3AX

4 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable, GSR Used for Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Truth Table

Table 782:

INPUTS					OUTPUTS	
D[0:3]	SD	CI	SP	CK	CO	Q[0:3]
D[0:3]	1	0	1	↑	0	D[0:3]
D[0:3]	1	1	1	↑	*	D[0:3]
X	0	0	X	X	0	Q[0:3]
X	X	0	0	X	0	Q[0:3]
X	X	1	0	X	*	Q[0:3]
X	0	1	1	↑	*	count+1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

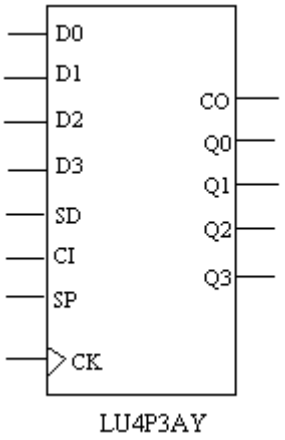
When GSR=0, CO=0, Q[0:3]=0 (D[0:3]=SP=CK=SD=X)

LU4P3AY

4 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable, GSR Used for Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Truth Table

Table 783:

INPUTS					OUTPUTS	
D[0:3]	SD	CI	SP	CK	CO	Q[0:3]
D[0:3]	1	0	1	↑	0	D[0:3]
D[0:3]	1	1	1	↑	*	D[0:3]
X	0	0	X	X	0	Q[0:3]
X	X	0	0	X	0	Q[0:3]
X	X	1	0	X	*	Q[0:3]
X	0	1	1	↑	*	count+1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

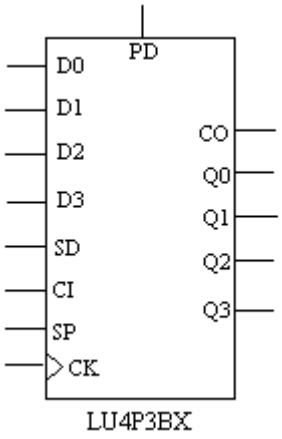
When GSR=0, CO=CI, Q[0:3]=1 (D[0:3]=CI=SP=CK=SD=X)

LU4P3BX

4 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Asynchronous Preset

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Truth Table

Table 784:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	PD	CO	Q[0:3]
X	X	0	X	X	1	0	1
X	X	1	X	X	1	1	1
D[0:3]	1	0	1	↑	0	0	D[0:3]
D[0:3]	1	1	1	↑	0	*	D[0:3]
X	0	0	X	X	0	0	Q[0:3]
X	X	0	0	X	0	0	Q[0:3]
X	X	1	0	X	0	*	Q[0:3]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

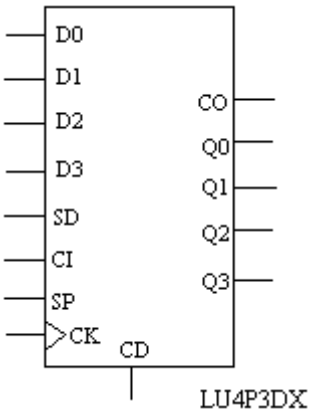
When GSR=0, CO=CI, Q[0:3]=1 (D[0:3]=SP=CK=SD=PD=X)

LU4P3DX

4 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Asynchronous Clear

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Truth Table

Table 785:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	CD	CO	Q[0:3]
X	X	X	X	X	1	0	0
D[0:3]	1	0	1	↑	0	0	D[0:3]
D[0:3]	1	1	1	↑	0	*	D[0:3]
X	0	0	X	X	0	0	Q[0:3]
X	X	0	0	X	0	0	Q[0:3]
X	X	1	0	X	0	*	Q[0:3]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

When GSR=0, CO=0, Q[0:3]=0 (D[0:3]=SP=CK=SD=CD=X)

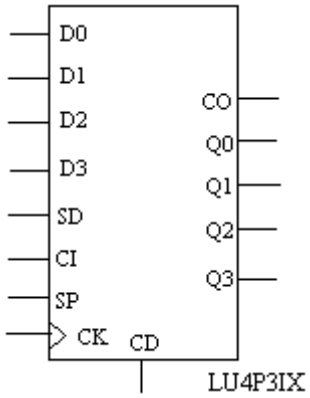
LU4P3IX

4 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Synchronous Clear (Clear overrides Enable)

Architectures Supported:

- LatticeECP/EC

- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, CD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Truth Table

Table 786:

INPUTS						**OUTPUTS**	
D[0:3]	SD	CI	SP	CK	CD	CO	Q[0:3]
X	X	X	X	↑	1	0	0
D[0:3]	1	0	1	↑	0	0	D[0:3]
D[0:3]	1	1	1	↑	0	*	D[0:3]
X	0	0	X	X	0	0	Q[0:3]
X	X	0	0	X	0	0	Q[0:3]
X	X	1	0	X	0	*	Q[0:3]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

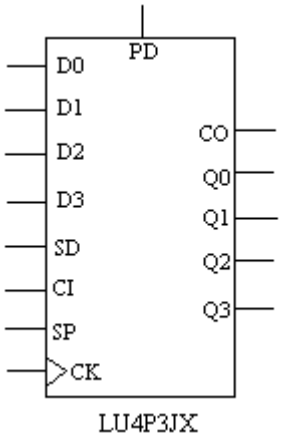
When GSR=0, CO=0, Q[0:3]=0 (D[0:3]=SP=CK=SD=CD=X)

LU4P3JX

4 Bit Positive Edge Triggered Loadable Up-Counter with Positive Clock Enable and Positive Level Synchronous Preset (Preset overrides Enable)

Architectures Supported:

- LatticeECP/EC
- LatticeSC/M
- LatticeXP



INPUTS: D0, D1, D2, D3, CI, SP, CK, SD, PD

OUTPUTS: CO, Q0, Q1, Q2, Q3

Truth Table

Table 787:

INPUTS						OUTPUTS	
D[0:3]	SD	CI	SP	CK	PD	CO	Q[0:3]
X	X	0	X	↑	1	0	1
X	X	1	X	↑	1	1	1
D[0:3]	1	0	1	↑	0	0	D[0:3]
D[0:3]	1	1	1	↑	0	*	D[0:3]
X	0	0	X	X	0	0	Q[0:3]
X	X	0	0	X	0	0	Q[0:3]
X	X	1	0	X	0	*	Q[0:3]
X	0	1	1	↑	0	*	count+1

X = Don’t care

* When Q[0:3] is 1111, CO will be 1; otherwise, CO will be 0

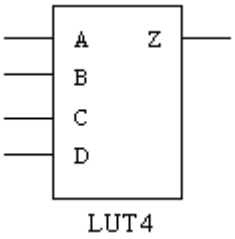
When GSR=0, CO=CI, Q[0:3]=1 (D[0:3]=SP=CK=SD=PD=X)

LUT4

4-Input Look Up Table

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: A, B, C, D

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 16'h0000)

Description

LUT4 defines the programmed state of a LUT4 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT4 programming. The contents of the look up table are addressed by the 4 input pins to access 1 of 16 locations.

The programming of the LUT4 (that is, the 0 or 1 value of each memory location within the LUT4) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For example, hex value BF80 produces these 16 memory locations and values:

1011 1111 1000 0000

Memory location 0 (D=0, C=0, B=0, A=0) contains a 0, memory location 2 (D=0, C=0, B=1, A=0) contains a 0. Memory location 15 (D=1, C=1, B=1, A=1) contains a 1, etc.

The LUT4 may encode the Boolean logic for any Boolean expression of 4 input variables. For example, if the required expression was:

Z = (D*C) + (B*!A)

then the INIT value can be derived from the truth table resulting from the expression:

```
D C B A : Z
0 0 0 0 : 0
0 0 0 1 : 0
0 0 1 0 : 1
0 0 1 1 : 0

0 1 0 0 : 0
```

```
0 1 0 1 : 0
0 1 1 0 : 1
0 1 1 1 : 0

1 0 0 0 : 0
1 0 0 1 : 0
1 0 1 0 : 1
1 0 1 1 : 0

1 1 0 0 : 1
1 1 0 1 : 1
1 1 1 0 : 1
1 1 1 1 : 1
```

INIT = F444 (16)

LUT4 Usage with Verilog HDL

```
// LUT4 module instantiation
LUT4
   #(.init  (16'hF444))
   I1 ( .A (A), .B (B), .C (C), .D (D), .Z (Q[0]) );
```

LUT4 Usage with VHDL

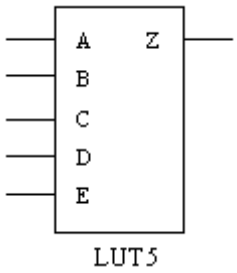
```
-- LUT4 component instantiation
I1 : LUT4
   Generic Map (INIT=>b"1111_0100_0100_0100")
   Port Map ( A=>A, B=>B, C=>C, D=>D, Z=>N_1 );
```

LUT5

5-Input Look Up Table

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: A, B, C, D, E

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 32'h00000000)

Description

LUT5 defines the programmed state of a LUT5 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT5 programming. The contents of the look up table are addressed by the 5 input pins to access 1 of 32 locations.

The programming of the LUT5 (that is, the 0 or 1 value of each memory location within the LUT5) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

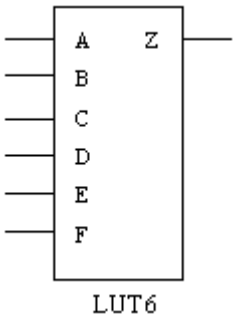
For more information on INIT attribute usage, see the LUT4 topic.

LUT6

6-Input Look Up Table

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: A, B, C, D, E, F

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 64'h0000000000000000)

Description

LUT6 defines the programmed state of a LUT6 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT6 programming. The contents of the look up table are addressed by the 6 input pins to access 1 of 64 locations.

The programming of the LUT6 (that is, the 0 or 1 value of each memory location within the LUT6) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For more information on INIT attribute usage, see the LUT4 topic.

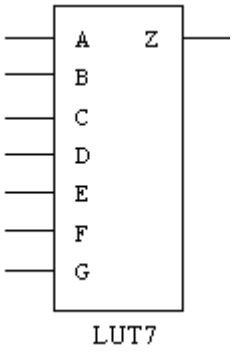
LUT7

7-Input Look Up Table

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L

- Platform Manager 2



INPUTS: A, B, C, D, E, F, G

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 128'00000000000000000000000000000000)

Description

LUT7 defines the programmed state of a LUT7 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT7 programming. The contents of the look up table are addressed by the 7 input pins to access 1 of 128 locations.

The programming of the LUT7 (that is, the 0 or 1 value of each memory location within the LUT7) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For more information on INIT attribute usage, see the LUT4 topic.

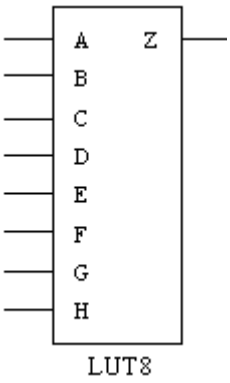
LUT8

8-Input Look Up Table

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: A, B, C, D, E, F, G, H

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 256'h00000000000000000000000000000000000000000000000000000000000000000)

Description

LUT8 defines the programmed state of a LUT8 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT8 programming. The contents of the look up table are addressed by the 8 input pins to access 1 of 256 locations.

The programming of the LUT8 (that is, the 0 or 1 value of each memory location within the LUT8) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For more information on INIT attribute usage, see the LUT4 topic.

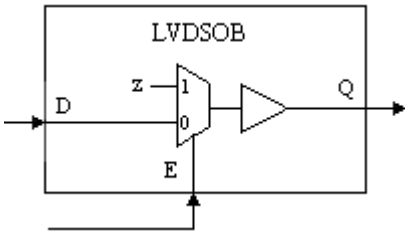
LVDSOB

LVDS Output Buffer

Architectures Supported:

- ECP5

- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, E

OUTPUT: Q

Description

The LVDSOB primitive is used to support post-PAR simulation of Dynamic Bank Controller. The Dynamic Bank Controller signals to the IO are hardwired and cannot be changed for the simulation, so the LVDSOB and INRDB primitives are defined to support the simulation.

For more information, refer to the following technical note on the Lattice web site:

- TN1198 - Power Estimation and Management for MachXO2 Device

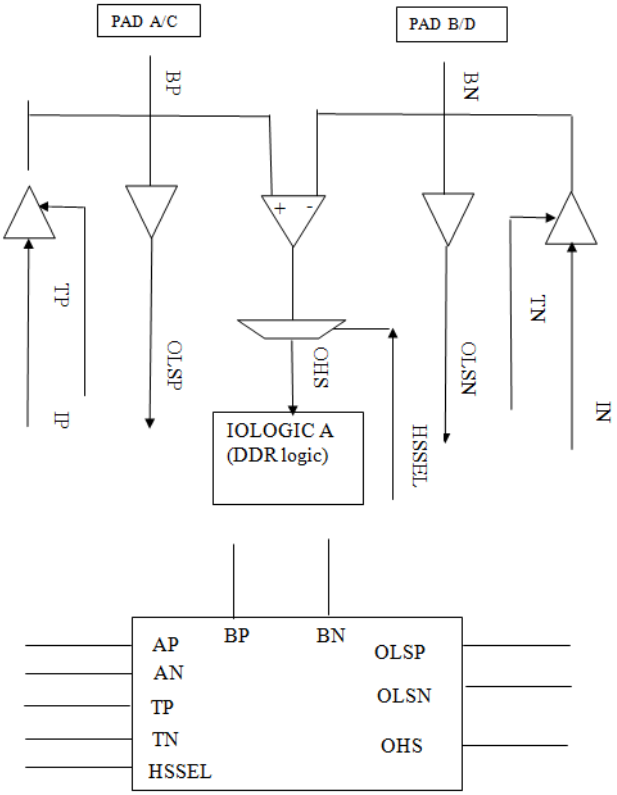
M

MIPI

Special Primitive for MIPI Input Support

Architectures Supported:

- LIFMD
- LIFMDF



INPUTS/OUTPUTS: BP, BN

INPUTS: AP, AN, HSSEL, TP, TN

OUTPUT: OLSP, OLSN, OHS

Description

This primitive is used when implementing MIPI interface. HS RX, LP RX and LP TX modes are supported.

The following are descriptions of MIPI port functions.

Table 788:

Port	I/O	Description
BP	I/O	PAD A, C
BN	I/O	PAD B, D
AP	I	Input from fabric to PAD A,C
AN	i	Input from fabric to PAD B,D
HSSEL	i	High Speed Select Signal. HSSEL=1: High Speed mode, 100 ohm differential termination is on. HS_SEL=0: Low Speed mode, 100 ohm termination is turned off.
TP	i	Tristate for PAD A,C
TN	I	Tristate for PAD B,D
OLSP	O	Low Speed to IOLOGIC A
OLSN	O	Low Speed to IOLOGIC B
OHS	O	High Speed Differential to IOLOGIC

MIPIDPHYA

Architectures Supported:

- LIFMD
- LIFMDF

INPUTS/OUTPUTS: CKP; CKN. DPy; DNy

INPUTS: CLKRXHSEN;CLKRXLPEN; CLKCDEN; CLKDTXLPP; CLKTXLPEN; CLKDTXLPN; CLKTXHSEN; CLKTXHSGATE; CLKTXHSPD; DyRXLPEN; DyCDEN; DyDTXLPP; DyTXLPEN; DyDTXLPN; DyRXHSEN; DyHSDESEREN; DyTXHSEN; DyHSTXDATA15; input DyHSTXDATA14, DYHSTXDATA13, DYHSTXDATA12, DYHSTXDATA11, DYHSTXDATA10; DyHSTXDATA9, DYHSTXDATA8, DYHSTXDATA7, DYHSTXDATA6, DYHSTXDATA5; DyHSTXDATA4, DYHSTXDATA3, DYHSTXDATA2, DYHSTXDATA1, DYHSTXDATA0; DyHSSEREN; DyTXHSPD; LBEN; PDDPHY; PDBIAS; PDCKG; CLKREF; PDPLL

OUTPUTS: CLKHSBYTE; CLKDRXLPP; CLKDRXLPN; CLKDCDN; CLKDRXHS; DyDRXLPP; DyDRXLPN; DyDCDP; DyDCDN; DyHSRXDATA15; DyHSRXDATA14, DYHSRXDATA13, DYHSRXDATA12, DYHSRXDATA11, DYHSRXDATA10; DyHSRXDATA9, DYHSRXDATA8, DYHSRXDATA7, DYHSRXDATA6, DYHSRXDATA5; DyHSRXDATA4, DYHSRXDATA3, DYHSRXDATA2, DYHSRXDATA1, DYHSRXDATA0;

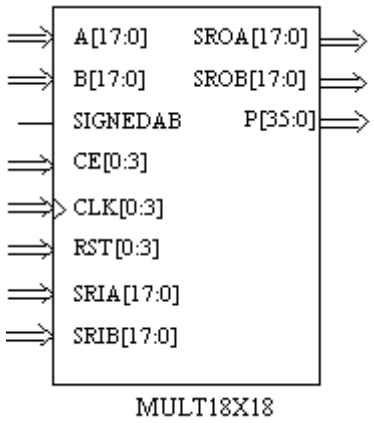
HSBYTECLKD; HSBYTECLKS; DySYNC; DyERRSYNC; DyNOSYNC; DyDRXHS; LOCK

MULT18X18

DSP Multiplier

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDAB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, P35, P34, P33, P32, P31, P30, P29, P28, P27, P26, P25, P24, P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A: "FALSE" (default), "TRUE"

SHIFT_IN_B: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

LatticeECP DSP Block Multiplier. MULT18X18 is a combinational signed 18-bit by 18-bit multiplier used in the DSP block. The value represented in the 18-bit input A is multiplied by the value represented in the 18-bit input B. Output P is the 36-bit product of A and B. MULT18X18 may be represented as either unsigned or two's complement signed. The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

There are 12 register controls signals that enter the DSP block: CLK[0:3], CE0[0:3], and RST[0:3]. Each incoming signal also has the option of being tied high, tied low, or inverted. These 12 control signals are used to control the register banks in the DSP block. Dynamic control signals must match the register pipelining of the datapath. To facilitate this, the following bank control for each individual dynamic signal's input register and pipeline register (total 8 signals x 2 registers / signal = 16 registers) is as follows:

- Bypass or no-bypass of the registers.
- Clock is selected from CLK[0:3], one of four sources available to the DSP block.
- Clock enable is selected from CE0[0:3], one of four sources available to the DSP block.
- Reset is selected from RST[0:3], one of four sources available to the DSP block.

You can turn registers off or on via attribute settings. For example, setting REG_INPUTA_CLK= "CLK0" means the input A register is used, and the clock drive A register is coming from CLK0 of the DSP block. Setting REG_INPUTA_CE= "CE1" means input A register CE control is coming from CE1 of the DSP block. Setting REG_INPUTA_RST= "RST3" means input A register reset control is coming from RST3 of the DSP block. If REG_INPUT_A_CLK="NONE", this means the input A register is bypassed, therefore, REG_INPUT_A_RST or REG_INPUT_A_CE becomes irrelevant.

In case you want to use the register but do not care about the clock enable (CE), then this pin needs to be tied to VCC (always enabled). In this case you could set REG_INPUT_A_CE="CE3", then tie CE3 of the to VCC. If you want to use the register but do not care about the reset (RST), then this pin must to be tied to GND (always do not reset). In this case, you you could set REG_INPUT_A_RST="RST2", then tie RST2 of the to GND.

SIGNEDAB is a pin which controls whether the multiplier performs the signed or unsigned operation. It applied to both operand A and B. It can be tied to VCC (signed) or GROUND (unsigned). There are also two delay registers associated with this control pin, in order to match with incoming data. Setting REG_SIGNEDAB_0_CLK= "CLK0 | CLK1 | CLK2 | CLK3"will turn on the pipeline register for SIGNEDAB. Setting REG_SIGNEDAB_0_CLK= "NONE" will turn off the first pipeline register for SIGNEDAB. Setting REG_SIGNEDAB_1_CLK= "NONE" will turn off the second pipeline register for SIGNEDAB.

Input registers receive operand values from a serial shift chain or routing input. There is separate control for A and B operands. When in shift chain mode, multiplier operands may be bypassed using the bank bypass feature. The shift chain supports one chain of two 18-bit operands or two chains of two 9-bit operands. GSR "DISABLED" attribute disables the asynchronous global set reset input when in user mode.

You can refer to the following technical note on the Lattice web site for more details.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT18X18 pin functions:

Table 789:

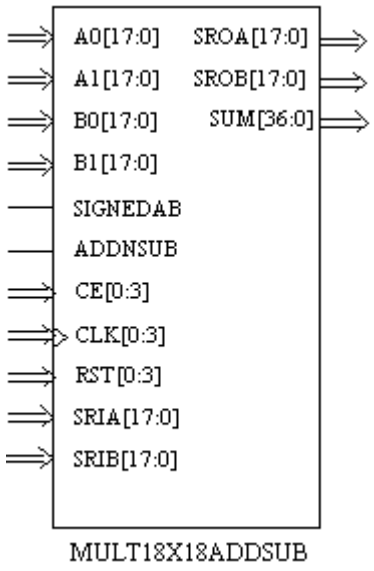
Function	Pins
input data A and B	A[17:0], B[17:0]0
signed input	SIGNEDAB
clock enable	CE[0:3]
clock input	CLK[0:3]
reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[17:0], SRIB[17:0]
shifted output A and B (from previous stage)	SROA[17:0], SROB[17:0]
output product data	P[35:0]

MULT18X18ADDSUB

ECP DSP Adder/Subtractor

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A017, A016, A015, A014, A013, A012, A011, A010, A09, A08, A07, A06, A05, A04, A03, A02, A01, A00, A117, A116, A115, A114, A113, A112, A111, A110, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, B017, B016, B015, B014, B013, B012, B011, B010, B09, B08, B07, B06, B05, B04, B03, B02, B01, B00, B117, B116, B115, B114, B113, B112, B111, B110, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, SIGNEDAB, ADDNSUB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM36, SUM35, SUM34, SUM33, SUM32, SUM31, SUM30, SUM29, SUM28, SUM27, SUM26, SUM25, SUM24, SUM23, SUM22, SUM21, SUM20, SUM19, SUM18, SUM17, SUM16, SUM15, SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB01_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A0: "FALSE" (default), "TRUE"

SHIFT_IN_B0: "FALSE" (default), "TRUE"

SHIFT_IN_A1: "FALSE" (default), "TRUE"

SHIFT_IN_B1: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

LatticeECP Block Adder/Subtractor. MULT18X18ADDSUB can be configured to either add or subtract its inputs, adding or subtracting the inputs from two multiplier products. The add/subtract control is either configured as a static HIGH (Vcc), LOW (GND). In Lattice Diamond, the static settings are implemented by setting Vcc or GND in the CIB ISB.

The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands

- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks and attributes.

You can also refer to the following technical note on the Lattice web site for more details.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT18X18ADDSUB pin functions:

Table 790:

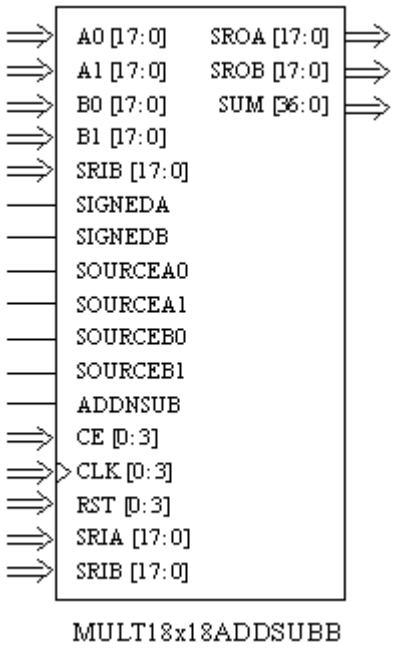
Function	Pins
input data A and B	A0|1[17:0], B0|1[17:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB
add/subtract (0 = add, 1 = subtract)	ADDNSUB
clock enable	CE[0:3]
clock input	CLK[0:3]
reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[17:0], SRIB[17:0]
shifted output A and B (from previous stage)	SROA[17:0], SROB[17:0]
output product sum data	SUM[36:0]

MULT18X18ADDSUBB

DSP Multiplier Add/Subtract

Architectures Supported:

- LatticeECP2/M
- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A017, A016, A015, A014, A013, A012, A011, A010, A09, A08, A07, A06, A05, A04, A03, A02, A01, A00, A117, A116, A115, A114, A113, A112, A111, A110, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, B017, B016, B015, B014, B013, B012, B011, B010, B09, B08, B07, B06, B05, B04, B03, B02, B01, B00, B117, B116, B115, B114, B113, B112, B111, B110, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, SIGNEDA, SIGNEDB, SOURCEA0, SOURCEA1, SOURCEB0, SOURCEB1, ADDNSUB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM36, SUM35, SUM34, SUM33, SUM32, SUM31, SUM30, SUM29, SUM28, SUM27, SUM26, SUM25, SUM24, SUM23, SUM22, SUM21, SUM20, SUM19, SUM18, SUM17, SUM16, SUM15, SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

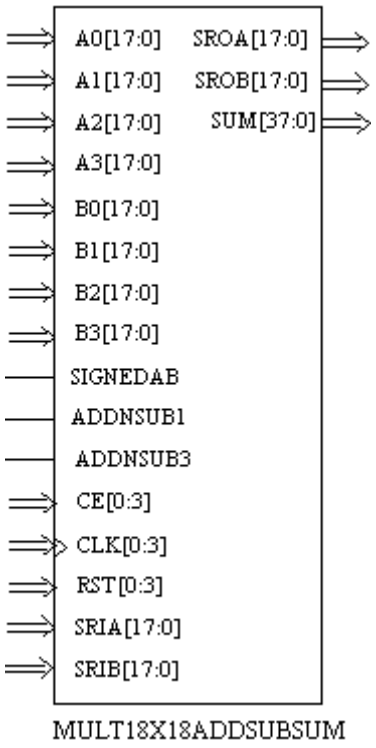
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT18X18ADDSUBSUM

ECP DSP Adder/Subtractor/Sum

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A017, A016, A015, A014, A013, A012, A011, A010, A09, A08, A07, A06, A05, A04, A03, A02, A01, A00, A117, A116, A115, A114, A113, A112, A111, A110, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, A217, A216, A215, A214, A213, A212, A211, A210, A29, A28, A27, A26, A25, A24, A23, A22, A21, A20, A317, A316, A315, A314, A313, A312, A311, A310, A39, A38, A37, A36, A35, A34, A33, A32, A31, A30, B017, B016, B015, B014, B013, B012, B011, B010, B09, B08, B07, B06, B05, B04, B03, B02, B01, B00, B117, B116, B115, B114, B113, B112, B111, B110, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, B217, B216, B215, B214, B213, B212, B211, B210, B29, B28, B27, B26, B25, B24, B23, B22, B21, B20, B317, B316, B315, B314, B313, B312, B311, B310, B39, B38, B37, B36, B35, B34, B33, B32, B31, B30, SIGNEDAB, ADDNSUB1, ADDNSUB3, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM37, SUM36, SUM35, SUM34, SUM33, SUM32, SUM31, SUM30, SUM29, SUM28, SUM27, SUM26, SUM25, SUM24, SUM23, SUM22, SUM21, SUM20,

SUM19, SUM18, SUM17, SUM16, SUM15, SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE0_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE1_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A0: "FALSE" (default), "TRUE"

SHIFT_IN_B0: "FALSE" (default), "TRUE"

SHIFT_IN_A1: "FALSE" (default), "TRUE"

SHIFT_IN_B1: "FALSE" (default), "TRUE"

SHIFT_IN_A2: "FALSE" (default), "TRUE"

SHIFT_IN_B2: "FALSE" (default), "TRUE"

SHIFT_IN_A3: "FALSE" (default), "TRUE"

SHIFT_IN_B3: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

LatticeECP DSP Block Adder/Subtractor. MULT18X18ADDSUBSUM can be configured to either add or subtract its inputs, adding or subtracting the inputs from two multiplier products. The add/subtract control is either configured as a static HIGH (Vcc), LOW (GND), or as dynamic control signals ADDNSUB1 and ADDNSUB3. In Lattice Diamond, the static settings are implemented by setting ADDNSUB1 and ADDNSUB3 signal to Vcc or GND in the CIB ISB.

The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks and attributes.

Refer to the following technical note on the Lattice web site.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT18X18ADDSUBSUM pin functions:

Table 791:

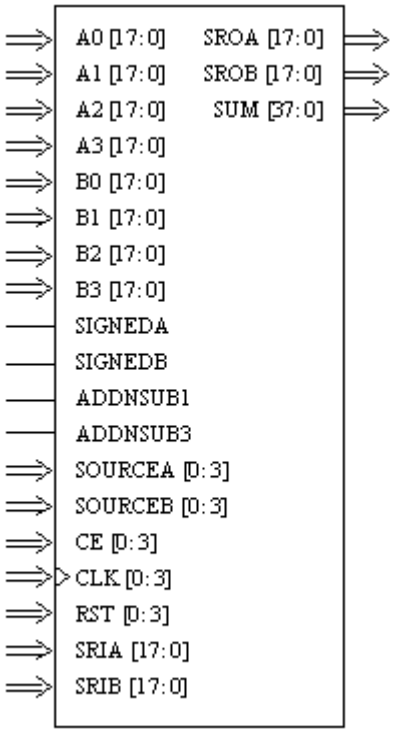
Function	Pins
input data A and B	A0|1|2|3[17:0], B0|1|2|3[17:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB
add/subtract (0 = add, 1 = subtract)	ADDNSUB1, ADDNSUB3
clock enable	CE[0:3]
clock input	CLK[0:3]
reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[17:0], SRIB[17:0]
shifted output A and B (from previous stage)	SROA[17:0], SROB[17:0]
output product sum data	SUM[37:0]

MULT18X18ADDSUBSUMB

DSP Multiplier Add/Subtract/Sum

Architectures Supported:

- LatticeECP2/M
- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A017, A016, A015, A014, A013, A012, A011, A010, A09, A08, A07, A06, A05, A04, A03, A02, A01, A00, A117, A116, A115, A114, A113, A112, A111, A110, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, A217, A216, A215, A214, A213, A212, A211, A210, A29, A28, A27, A26, A25, A24, A23, A22, A21, A20, A317, A316, A315, A314, A313, A312, A311, A310, A39, A38, A37, A36, A35, A34, A33, A32, A31, A30, B017, B016, B015, B014, B013, B012, B011, B010, B09, B08, B07, B06, B05, B04, B03, B02, B01, B00, B117, B116, B115, B114, B113, B112, B111, B110, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, B217, B216, B215, B214, B213, B212, B211, B210, B29, B28, B27, B26, B25, B24, B23, B22, B21, B20, B317, B316, B315, B314, B313, B312, B311, B310, B39, B38, B37, B36, B35, B34, B33, B32, B31, B30, SIGNEDA, SIGNEDB, ADDNSUB1, ADDNSUB3, SOURCEA0, SOURCEA1, SOURCEA2, SOURCEA3, SOURCEB0, SOURCEB1, SOURCEB2, SOURCEB3, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM37, SUM36, SUM35, SUM34, SUM33, SUM32, SUM31, SUM30, SUM29, SUM28, SUM27, SUM26, SUM25, SUM24, SUM23, SUM22, SUM21, SUM20,

SUM19, SUM18, SUM17, SUM16, SUM15, SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

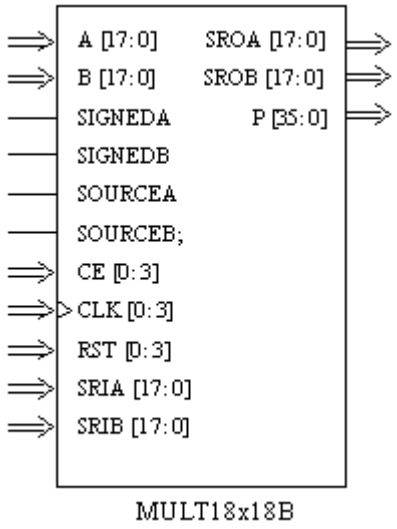
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT18X18B

DSP Multiplier

Architectures Supported:

- LatticeECP2/M
- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDA, SIGNEDB, SOURCEA, SOURCEB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, P35, P34, P33, P32, P31, P30, P29, P28, P27, P26, P25, P24, P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

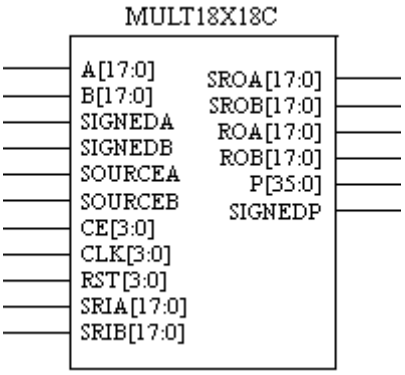
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT18X18C

DSP Multiplier

Architectures Supported:

- ECP5
- LatticeECP3



INPUTS: A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDA, SIGNEDB, SOURCEA, SOURCEB, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, ROA17, ROA16, ROA15, ROA14, ROA13, ROA12, ROA11, ROA10, ROA9, ROA8, ROA7, ROA6, ROA5, ROA4, ROA3, ROA2, ROA1, ROA0, ROB17, ROB16, ROB15, ROB14, ROB13, ROB12, ROB11, ROB10, ROB9, ROB8, ROB7, ROB6, ROB5, ROB4, ROB3, ROB2, ROB1, ROB0, P35, P34, P33, P32, P31, P30, P29, P28, P27, P26, P25, P24, P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0, SIGNEDP

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

CAS_MATCH_REG: "FALSE" (default), "TRUE"

MULT_BYPASS: "DISABLED" (default), "ENABLED"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

MULT18X18C Port Description

Table 792:

I/O	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	A	A	Bus	17:0	A input
I	B	B	Bus	17:0	B input
I	SRIA	SRIA	Bus	17:0	Shift input A
I	SRIB	SRIB	Bus	17:0	Shift input B
I	SIGNEDA	SIGNEDA	Bit	N/A	Input A sign selection
I	SIGNEDB	SIGNEDB	Bit	N/A	Input B sign selection
I	SOURCEA	SOURCEA	Bit	N/A	Source A selection
I	SOURCEB	SOURCEB	Bit	N/A	Source B selection
I	CLK0	CLK0	Bit	N/A	Clock Input
I	CLK1	CLK1	Bit	N/A	Clock Input
I	CLK2	CLK2	Bit	N/A	Clock Input
I	CLK3	CLK3	Bit	N/A	Clock Input
I	CE0	CE0	Bit	N/A	Clock Enable Input
I	CE1	CE1	Bit	N/A	Clock Enable Input
I	CE2	CE2	Bit	N/A	Clock Enable Input
I	CE3	CE3	Bit	N/A	Clock Enable Input
I	RST0	RST0	Bit	N/A	Reset Input
I	RST1	RST1	Bit	N/A	Reset Input
I	RST2	RST2	Bit	N/A	Reset Input

Table 792:

I/O	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	RST3	RST3	Bit	N/A	Reset Input
O	P	P	Bus	35:0	Product
O	SROA	SROA	Bus	17:0	Shift A Output
O	SROB	SROB	Bus	17:0	Shift B Output
O	ROA	ROA	Bus	17:0	Registered Output A from Multiplier
O	ROB	ROB	Bus	17:0	Registered Output B from Multiplier
O	SIGNEDP	SIGNEDP	Bit	N/A	Output Sign Bit (result of SignedA or SignedB)

MULT18X18C Attribute Description

Table 793:

Name	Value	Default	Description
REG_INPUTA_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input A clock selection
REG_INPUTA_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input A clock enable selection
REG_INPUTA_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input A reset selection
REG_INPUTB_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input B clock selection
REG_INPUTB_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input B clock enable selection
REG_INPUTB_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input B reset selection
REG_PIPELINE_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Pipeline clock selection
REG_PIPELINE_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Pipeline clock enable selection
REG_PIPELINE_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Pipeline reset selection
REG_OUTPUT_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Output clock selection
REG_OUTPUT_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Output clock enable selection
REG_OUTPUT_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Output reset selection

Table 793:

Name	Value	Default	Description
CAS_MATCH_REG	"FALSE", "TRUE"	"FALSE"	Cascade match register option
MULT_BYPASS	"DISABLED", "ENABLED"	"ENABLED"	Multiplier bypass option
RESETMODE	"SYNC", "ASYNC"	"SYNC"	Global set reset selection
GSR	"ENABLED", "DISABLED"	"ENABLED"	Reset mode selection

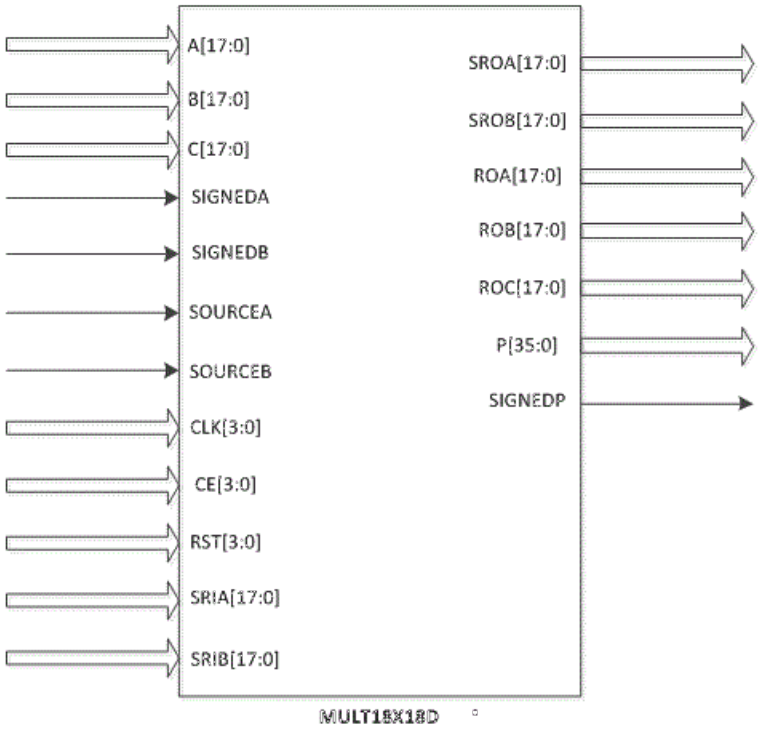
Note:
The multipliers linked by SROA/SRIA or SROB/SRIB pin pair can only be implemented in a continuous shift chain with multipliers placed next to each other. However, the length of the shift chain is limited on each DSP row and different depending on the device selected. For example, with the LatticeECP3-35 device, each DSP row has 32 MULT18s. So for the shift chained MULT18, it can support a continuous chain of length 32 for MULT18. If a chain is more than 32 MULT18s, you have to employ SRO/A,B pin pair for connecting the 32nd and 33rd MULT18s for implementing the chain in two DSP rows. But this capability is only limited to port A on SRO side in LatticeECP3. That is to say, only SROA is allowed to connect to A,B input pin of another MULT18, but not SROB. So if a chain is linked by SROB/SRIB pin pair, the length cannot exceed 32 for the LatticeECP3-35 device.

MULT18X18D

DSP Multiplier

Architectures Supported:

- ECP5



INPUTS: A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, C17, C16, C15, C14, C13, C12, C11, C10, C9, C8, C7, C6, C5, C4, C3, C2, C1, C0, SIGNEDA, SIGNEDB, SOURCEA, SOURCEB, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, ROA17, ROA16, ROA15, ROA14, ROA13, ROA12, ROA11, ROA10, ROA9, ROA8, ROA7, ROA6, ROA5, ROA4, ROA3, ROA2, ROA1, ROA0, ROB17, ROB16, ROB15, ROB14, ROB13, ROB12, ROB11, ROB10, ROB9, ROB8, ROB7, ROB6, ROB5, ROB4, ROB3, ROB2, ROB1, ROB0, ROC17, ROC16, ROC15, ROC14, ROC13, ROC12, ROC11, ROC10, ROC9, ROC8, ROC7, ROC6, ROC5, ROC4, ROC3, ROC2, ROC1, ROC, 0P35, P34, P33, P32, P31, P30, P29, P28, P27, P26, P25, P24, P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0, SIGNEDP

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTC_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTC_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTC_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

CLK0_DIV: “ENABLED” (default) “DISABLED”

CLK1_DIV: “ENABLED” (default) “DISABLED”

CLK2_DIV: “ENABLED” (default) “DISABLED”

CLK3_DIV: “ENABLED” (default) “DISABLED”

CAS_MATCH_REG: "FALSE" (default), "TRUE"

MULT_BYPASS: "DISABLED" (default), "ENABLED"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

SOURCEB_MODE"B_SHIFT"
(default),"C_SHIFT","B_C_DYNAMIC","HIGHSPEED"

MULT18X18D Port Description

Table 794:

I/O	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	A	A	Bus	17:0	A input
I	B	B	Bus	17:0	B input
I	C	C	Bus	17:0	C input
I	SRIA	SRIA	Bus	17:0	Shift input A
I	SRIB	SRIB	Bus	17:0	Shift input B
I	SIGNEDA	SIGNEDA	Bit	N/A	Input A sign selection
I	SIGNEDB	SIGNEDB	Bit	N/A	Input B sign selection
I	SOURCEA	SOURCEA	Bit	N/A	Source A selection
I	SOURCEB	SOURCEB	Bit	N/A	Source B selection
I	CLK0	CLK0	Bit	N/A	Clock Input
I	CLK1	CLK1	Bit	N/A	Clock Input
I	CLK2	CLK2	Bit	N/A	Clock Input
I	CLK3	CLK3	Bit	N/A	Clock Input
I	CE0	CE0	Bit	N/A	Clock Enable Input
I	CE1	CE1	Bit	N/A	Clock Enable Input
I	CE2	CE2	Bit	N/A	Clock Enable Input
I	CE3	CE3	Bit	N/A	Clock Enable Input
I	RST0	RST0	Bit	N/A	Reset Input
I	RST1	RST1	Bit	N/A	Reset Input
I	RST2	RST2	Bit	N/A	Reset Input
I	RST3	RST3	Bit	N/A	Reset Input
O	P	P	Bus	35:0	Product
O	SROA	SROA	Bus	17:0	Shift A Output
O	SROB	SROB	Bus	17:0	Shift B Output
O	ROA	ROA	Bus	17:0	Registered Output A from Multiplier
O	ROB	ROB	Bus	17:0	Registered Output B from Multiplier
O	ROC	ROC	Bus	17:0	Registered Output C from Multiplier
O	SIGNEDP	SIGNEDP	Bit	N/A	Output Sign Bit (result of SignedA or SignedB)

MULT18X18D Attribute Description

Table 795:

Name	Value	Default	Description
REG_INPUTA_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input A clock selection
REG_INPUTA_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input A clock enable selection
REG_INPUTA_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input A reset selection
REG_INPUTB_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input B clock selection
REG_INPUTB_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input B clock enable selection
REG_INPUTB_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input B reset selection
REG_INPUTC_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input C clock selection
REG_INPUTC_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input C clock enable selection
REG_INPUTC_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input C reset selection
REG_PIPELINE_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Pipeline clock selection
REG_PIPELINE_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Pipeline clock enable selection
REG_PIPELINE_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Pipeline reset selection
REG_OUTPUT_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Output clock selection
REG_OUTPUT_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Output clock enable selection
REG_OUTPUT_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Output reset selection
CLK0_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK0 divider setting.
CLK1_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK1 divider setting.
CLK2_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK2 divider setting.
CLK3_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK3 divider setting.
HIGHSPEED_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	NONE"	High speed clock setting.
CAS_MATCH_REG	"FALSE", "TRUE"	"FALSE"	Cascade match register option

Table 795:

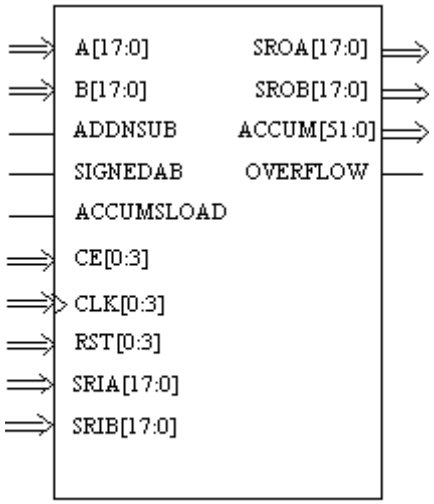
Name	Value	Default	Description
SOURCEB_MODE	"B_SHIFT", "C_SHIFT", "B_C_DYNAMIC", "HIGHSPEED"	"B_SHIFT"	SOURCEB mode.
MULT_BYPASS	"DISABLED", "ENABLED"	"ENABLED"	Multiplier bypass option
RESETMODE	"SYNC", "ASYNC"	"SYNC"	Global set reset selection
GSR	"ENABLED", "DISABLED"	"ENABLED"	Reset mode selection

MULT18X18MAC

ECP DSP Multiplier Accumulate

Architectures Supported:

- LatticeECP (DSP Blocks Only)



MULT18X18MAC

INPUTS: A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, ADDNSUB, SIGNEDAB, ACCUMSLOAD, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, ACCUM51, ACCUM50, ACCUM49, ACCUM48, ACCUM47, ACCUM46, ACCUM45, ACCUM44, ACCUM43, ACCUM42, ACCUM41, ACCUM40, ACCUM39, ACCUM38, ACCUM37, ACCUM36, ACCUM35, ACCUM34, ACCUM33, ACCUM32, ACCUM31, ACCUM30, ACCUM29, ACCUM28, ACCUM27, ACCUM26, ACCUM25, ACCUM24, ACCUM23, ACCUM22, ACCUM21, ACCUM20, ACCUM19, ACCUM18, ACCUM17, ACCUM16, ACCUM15, ACCUM14, ACCUM13, ACCUM12, ACCUM11, ACCUM10, ACCUM9, ACCUM8, ACCUM7, ACCUM6, ACCUM5, ACCUM4, ACCUM3, ACCUM2, ACCUM1, ACCUM0, OVERFLOW

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ACCUMSLOAD_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ACCUMSLOAD_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ACCUMSLOAD_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ACCUMSLOAD_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ACCUMSLOAD_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ACCUMSLOAD_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A: "FALSE" (default), "TRUE"

SHIFT_IN_B: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

MULT18X18MAC supports operand bit widths for 18X18 multiplication and accumulates the output up to 52 bits. The DSP block includes optional registers for the input and intermediate pipeline stage. Pipeline stages may be set using a pipeline attribute. The output registers are required for the accumulator. Signed and unsigned arithmetic are supported. The OVERFLOW bit is also provided when the accumulated results are in the overflow condition. ACCUMSLOAD determines the mode of operation for either loading the multiplier product or to accumulate.

The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks and attributes.

Refer to the following technical note on the Lattice web site.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT18X18MAC pin functions:

Table 796:

Function	Pins
input data A and B	A[17:0], B[17:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB

Table 796:

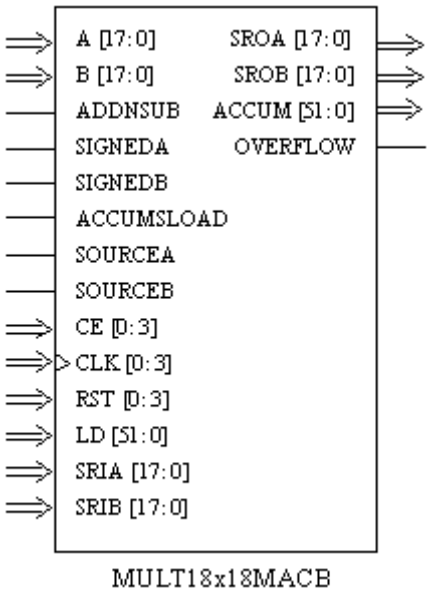
Function	Pins
add/subtract (0 = add, 1 = subtract)	ADDNSUB
Accumulate (HIGH) /Load (LOW) Mode	ACCUMSLOAD
clock enable	CE[0:3]
clock input	CLK[0:3]
clock reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[17:0], SRIB[17:0]
shifted output A and B (from previous stage)	SROA[17:0], SROB[17:0]
output data	ACCUM[51:0]
overflow	OVERFLOW

MULT18X18MACB

DSP Multiplier Accumulate

Architectures Supported:

- LatticeECP2/M
- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, ADDNSUB, SIGNEDA, SIGNEDB, ACCUMSLOAD, SOURCEA, SOURCEB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, LD51, LD50, LD49, LD48, LD47, LD46, LD45, LD44, LD43, LD42, LD41, LD40, LD39, LD38, LD37, LD36, LD35, LD34, LD33, LD32, LD31, LD30, LD29, LD28, LD27, LD26, LD25, LD24, LD23, LD22, LD21, LD20, LD19, LD18, LD17, LD16, LD15, LD14, LD13, LD12, LD11, LD10, LD9, LD8, LD7, LD6, LD5, LD4, LD3, LD2, LD1, LD0, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, ACCUM51, ACCUM50, ACCUM49, ACCUM48, ACCUM47, ACCUM46, ACCUM45, ACCUM44, ACCUM43, ACCUM42, ACCUM41, ACCUM40, ACCUM39, ACCUM38, ACCUM37, ACCUM36, ACCUM35, ACCUM34, ACCUM33, ACCUM32, ACCUM31, ACCUM30, ACCUM29, ACCUM28, ACCUM27, ACCUM26, ACCUM25, ACCUM24, ACCUM23, ACCUM22, ACCUM21, ACCUM20, ACCUM19, ACCUM18, ACCUM17, ACCUM16, ACCUM15, ACCUM14, ACCUM13, ACCUM12, ACCUM11, ACCUM10, ACCUM9, ACCUM8, ACCUM7, ACCUM6, ACCUM5, ACCUM4, ACCUM3, ACCUM2, ACCUM1, ACCUM0, OVERFLOW

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ACCUMSLOAD_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ACCUMSLOAD_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ACCUMSLOAD_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ACCUMSLOAD_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ACCUMSLOAD_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ACCUMSLOAD_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

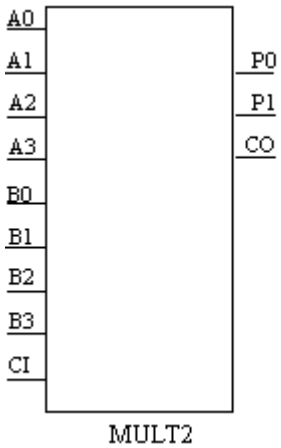
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT2

2x2 Multiplier

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeXP
- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A0, A1, A2, A3, B0, B1, B2, B3, CI

OUTPUTS: P0, P1, CO

Description

MULT2 is a 2x2 multiplier. This primitive is useful for implementing an array multiplier using a dedicated carry chain. With this unique configuration, the two separate “and” inputs can be added together to perform the add and shift operations within a single slice. MULT2 must be cascaded together when performing the multiply function. Here are descriptions of the MULT2 pins:

Table 797:

Function	Pins
multiplicand	A0, A1, A2, A3
multiplier	B0, B1, B2, B3
carry in	CI
carry out	CO
output	P0, P1

The equations for this primitive are shown in the table below:

Table 798:

Equations
CO_int, P0 = A0*B0 + A1*B1 + CI
CO, P1 = A2*B2 + A3*B3 + CO_int

CO_int and P0 are the carry-out and sum of a full adder with inputs (A0 AND B0), (A1 AND B1), and CI. CO and P1 are the carry-out and sum of a full adder with inputs (A2 AND B2), (A3 AND B3), and CO_int.

Refer to the following technical notes on the Lattice web site.

- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide
- TN1057 - LatticeECP sysDSP Usage Guide

Note

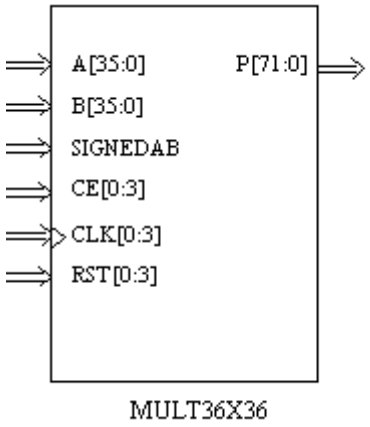
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

MULT36X36

ECP DSP Multiplier

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A35, A34, A33, A32, A31, A30, A29, A28, A27, A26, A25, A24, A23, A22, A21, A20, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B35, B34, B33, B32, B31, B30, B29, B28, B27, B26, B25, B24, B23, B22, B21, B20, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDAB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3

OUTPUTS: P71, P70, P69, P68, P67, P66, P65, P64, P63, P62, P61, P60, P59, P58, P57, P56, P55, P54, P53, P52, P51, P50, P49, P48, P47, P46, P45, P44, P43, P42, P41, P40, P39, P38, P37, P36, P35, P34, P33, P32, P31, P30, P29, P28, P27, P26, P25, P24, P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

LatticeECP Block Multiplier. MULT36X36 is a combinational signed 36-bit by 36-bit multiplier used in the DSP block. The value represented in the 36-bit input A is multiplied by the value represented in the 36-bit input B. Output P is the 72-bit product of A and B. MULT36X36 may be represented as either unsigned or two's complement signed. The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks and attributes.

Input registers receive operand values from a serial shift chain or routing input. There is separate control for A and B operands. When in shift chain mode, multiplier operands may be bypassed using the bank bypass feature. The shift chain supports one chain of two 18-bit operands or two chains of two 9-bit operands. GSR "DISABLED" attribute disables the asynchronous global set reset input when in user mode.

Refer to the following technical note on the Lattice web site.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT36X36 pin functions:

Table 799:

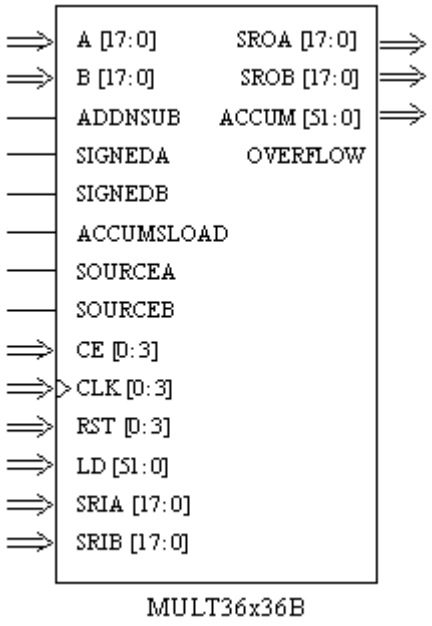
Function	Pins
input data A and B	A[35:0], B[35:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB
clock enable	CE[0:3]
clock input	CLK[0:3]
clock reset	RST[0:3]
output data	P[71:0]

MULT36X36B

DSP Multiplier

Architectures Supported:

- LatticeECP2/M
- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A35, A34, A33, A32, A31, A30, A29, A28, A27, A26, A25, A24, A23, A22, A21, A20, A19, A18, A17, A16, A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0, B35, B34, B33, B32, B31, B30, B29, B28, B27, B26, B25, B24, B23, B22, B21, B20, B19, B18, B17, B16, B15, B14, B13, B12, B11, B10, B9, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDA, SIGNEDB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3

OUTPUTS: P71, P70, P69, P68, P67, P66, P65, P64, P63, P62, P61, P60, P59, P58, P57, P56, P55, P54, P53, P52, P51, P50, P49, P48, P47, P46, P45, P44, P43, P42, P41, P40, P39, P38, P37, P36, P35, P34, P33, P32, P31, P30, P29, P28, P27, P26, P25, P24, P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

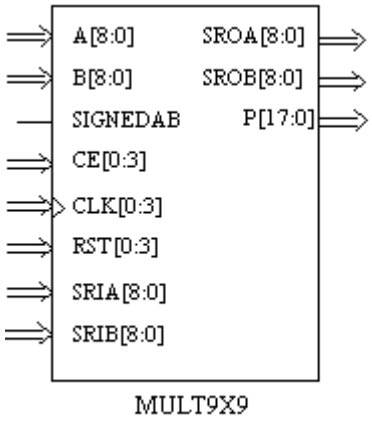
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT9X9

ECP DSP Multiplier

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A8, A7, A6, A5, A4, A3, A2, A1, A0, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDAB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3,

SROB2, SROB1, SROB0, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A: "FALSE" (default), "TRUE"

SHIFT_IN_B: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

LatticeECP DSP Block Multiplier. MULT9X9 is a combinational signed 9-bit by 9-bit multiplier used in the DSP block. The value represented in the 9-bit input A is multiplied by the value represented in the 9-bit input B. Output P is the

18-bit product of A and B. MULT9X9 may be represented as either unsigned or two's complement signed. The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks and attributes.

Refer to the following technical note on the Lattice web site.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT9X9 pin functions:

Table 800:

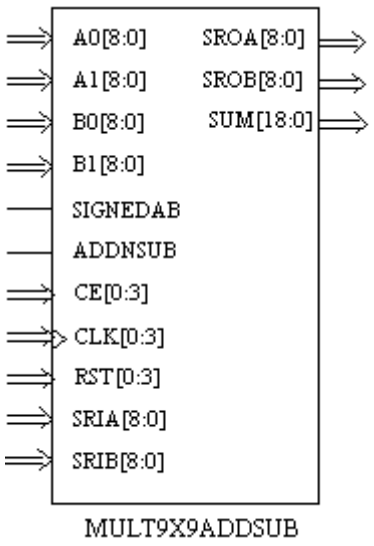
Function	Pins
input data A and B	A[8:0], B[8:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB
clock enable	CE[0:3]
clock input	CLK[0:3]
reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[8:0], SRIB[8:0]
shifted output A and B (from previous stage)	SROA[8:0], SROB[8:0]
output product data	P[17:0]

MULT9X9ADDSUB

ECP DSP Multiplier Add/Subtract

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A08, A07, A06, A05, A04, A03, A02, A01, A00, A18, A17, A16, A15, A14, A13, A12, A11, A10, B08, B07, B06, B05, B04, B03, B02, B01, B00, B18, B17, B16, B15, B14, B13, B12, B11, B10, SIGNEDAB, ADDNSUB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM18, SUM17, SUM16, SUM15, SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A0: "FALSE" (default), "TRUE"

SHIFT_IN_B0: "FALSE" (default), "TRUE"

SHIFT_IN_A1: "FALSE" (default), "TRUE"

SHIFT_IN_B1: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

LatticeECP DSP Block Adder/Subtractor. MULT18X18ADDSUB can be configured to either add or subtract its inputs, adding or subtracting the inputs from two multiplier products. The add/subtract control is either configured as a static HIGH (Vcc), LOW (GND), or as dynamic control signals ADDNSUB1 and ADDNSUB3. In Lattice Diamond, the static settings are implemented by setting ADDNSUB1 and ADDNSUB3 signal to Vcc or GND in the CIB ISB.

The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks and attributes.

Refer to the following technical note on the Lattice web site.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT9X9ADDSUB pin functions:

Table 801:

Function	Pins
input data A and B	A0|1[8:0], B0|1[8:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB
add/subtract (0 = add, 1 = subtract)	ADDNSUB
clock enable	CE[0:3]
clock input	CLK[0:3]
reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[8:0], SRIB[8:0]
shifted output A and B (from previous stage)	SROA[8:0], SROB[8:0]
output product sum data	SUM[18:0]

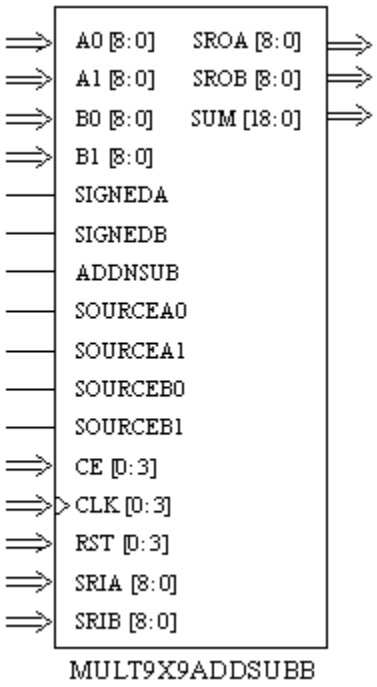
MULT9X9ADDSUBB

DSP Multiplier Add/Subtract

Architectures Supported:

- LatticeECP2/M

- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A08, A07, A06, A05, A04, A03, A02, A01, A00, A18, A17, A16, A15, A14, A13, A12, A11, A10, B08, B07, B06, B05, B04, B03, B02, B01, B00, B18, B17, B16, B15, B14, B13, B12, B11, B10, SIGNEDA, SIGNEDB, ADDNSUB, SOURCEA0, SOURCEA1, SOURCEB0, SOURCEB1, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM18, SUM17, SUM16, SUM15, SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

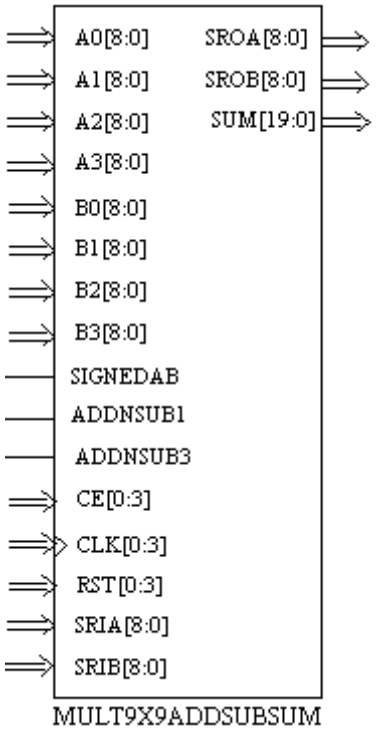
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT9X9ADDSUBSUM

ECP DSP Adder/Subtractor/Sum

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A08, A07, A06, A05, A04, A03, A02, A01, A00, A18, A17, A16, A15, A14, A13, A12, A11, A10, A28, A27, A26, A25, A24, A23, A22, A21, A20, A38, A37, A36, A35, A34, A33, A32, A31, A30, B08, B07, B06, B05, B04, B03, B02, B01, B00, B18, B17, B16, B15, B14, B13, B12, B11, B10, B28, B27, B26, B25, B24, B23, B22, B21, B20, B38, B37, B36, B35, B34, B33, B32, B31, B30, SIGNEDAB, ADDNSUB1, ADDNSUB3, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM19, SUM18, SUM17, SUM16, SUM15, SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE0_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE1_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A0: "FALSE" (default), "TRUE"

SHIFT_IN_B0: "FALSE" (default), "TRUE"

SHIFT_IN_A1: "FALSE" (default), "TRUE"

SHIFT_IN_B1: "FALSE" (default), "TRUE"

SHIFT_IN_A2: "FALSE" (default), "TRUE"

SHIFT_IN_B2: "FALSE" (default), "TRUE"

SHIFT_IN_A3: "FALSE" (default), "TRUE"

SHIFT_IN_B3: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

LatticeECP DSP Block Adder/Subtractor. MULT9X9ADDSUBSUM can be configured to either add or subtract its inputs, adding or subtracting the inputs from two multiplier products. The add/subtract control is either configured as a static HIGH (Vcc), LOW (GND), or as dynamic control signals ADDNSUB1 and ADDNSUB3. In Lattice Diamond, the static settings are implemented by setting ADDNSUB1 and ADDNSUB3 signal to Vcc or GND in the CIB ISB.

The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks and attributes.

Refer to the following technical note on the Lattice web site.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT9X9ADDSUBSUM pin functions:

Table 802:

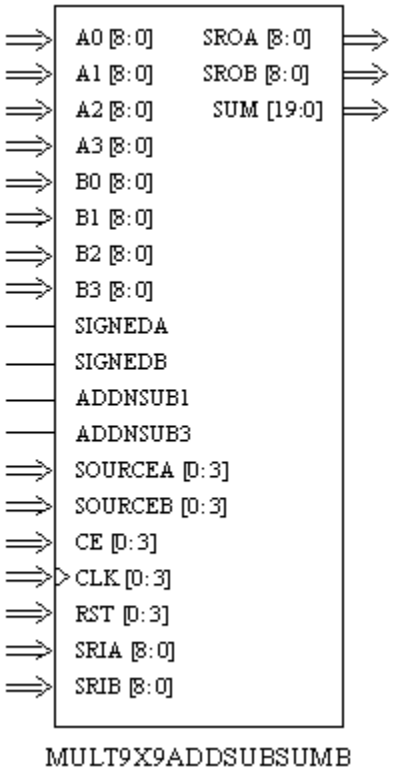
Function	Pins
input data A and B	A0|1|2|3[8:0], B0|1|2|3[8:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB
add/subtract (0 = add, 1 = subtract)	ADDNSUB1, ADDNSUB3
clock enable	CE[0:3]
clock input	CLK[0:3]
reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[8:0], SRIB[8:0]
shifted output A and B (from previous stage)	SROA[8:0], SROB[8:0]
output product sum data	SUM[19:0]

MULT9X9ADDSUBSUMB

DSP Multiplier Add/Subtract/Sum

Architectures Supported:

- LatticeECP2/M
- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A08, A07, A06, A05, A04, A03, A02, A01, A00, A18, A17, A16, A15, A14, A13, A12, A11, A10, A28, A27, A26, A25, A24, A23, A22, A21, A20, A38, A37, A36, A35, A34, A33, A32, A31, A30, B08, B07, B06, B05, B04, B03, B02, B01, B00, B18, B17, B16, B15, B14, B13, B12, B11, B10, B28, B27, B26, B25, B24, B23, B22, B21, B20, B38, B37, B36, B35, B34, B33, B32, B31, B30, SIGNEDA, SIGNEDB, ADDNSUB1, ADDNSUB3, SOURCEA0, SOURCEA1, SOURCEA2, SOURCEA3, SOURCEB0, SOURCEB1, SOURCEB2, SOURCEB3, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, SUM19, SUM18, SUM17, SUM16, SUM15,

SUM14, SUM13, SUM12, SUM11, SUM10, SUM9, SUM8, SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0

ATTRIBUTES:

REG_INPUTA0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTA3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE2_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE2_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE2_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE3_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE3_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE3_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB1_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB1_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB1_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ADDNSUB3_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ADDNSUB3_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ADDNSUB3_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

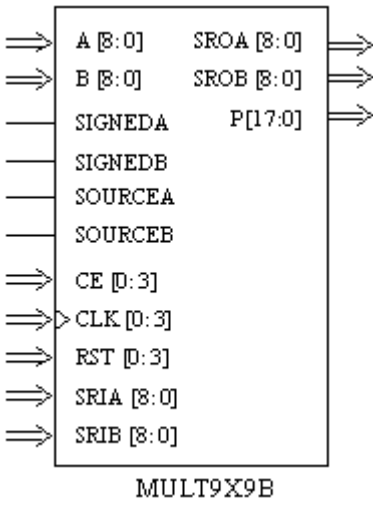
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT9X9B

DSP Multiplier

Architectures Supported:

- LatticeECP2/M
- LatticeECP3 (*for LatticeECP2/M backward compatibility only*)
- LatticeXP2



INPUTS: A8, A7, A6, A5, A4, A3, A2, A1, A0, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDA, SIGNEDB, SOURCEA, SOURCEB, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDB_RST: "RST0" (default), "RST1", "RST2", "RST3"

GSR: "ENABLED" (default), "DISABLED"

Description

Refer to the following technical notes on the Lattice web site.

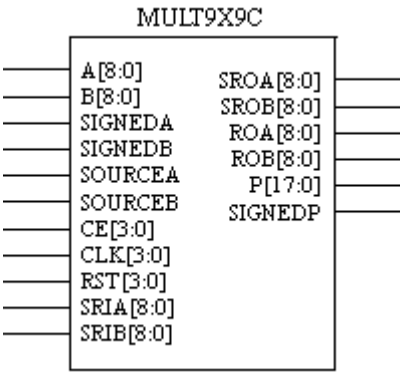
- TN1182 - LatticeECP3 sysDSP Usage Guide
- TN1107 - LatticeECP2/M sysDSP Usage Guide
- TN1140 - LatticeXP2 sysDSP Usage Guide

MULT9X9C

DSP Multiplier

Architectures Supported:

- ECP5
- LatticeECP3



INPUTS: A8, A7, A6, A5, A4, A3, A2, A1, A0, B8, B7, B6, B5, B4, B3, B2, B1, B0, SIGNEDA, SIGNEDB, SOURCEA, SOURCEB, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SRIA8, SRIA7,

SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, ROA8, ROA7, ROA6, ROA5, ROA4, ROA3, ROA2, ROA1, ROA0, ROB8, ROB7, ROB6, ROB5, ROB4, ROB3, ROB2, ROB1, ROB0, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0, SIGNEDP

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

CAS_MATCH_REG: "FALSE" (default), "TRUE"

MULT_BYPASS: "DISABLED" (default), "ENABLED"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

MULT9X9C Port Description

Table 803:

I/O	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	A	A[8:0]	Bus	8:0	A input
I	B	B[8:0]	Bus	8:0	B input
I	SRIA	SRIA[8:0]	Bus	8:0	Shift input A

Table 803:

I/O	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	SRIB	SRIB[8:0]	Bus	8:0	Shift input B
I	SIGNEDA	SIGNEDA	Bit	N/A	Input A sign selection
I	SIGNEDB	SIGNEDB	Bit	N/A	Input B sign selection
I	SOURCEA	SOURCEA	Bit	N/A	Source A selection
I	SOURCEB	SOURCEB	Bit	N/A	Source B selection
I	CLK0	CLK0	Bit	N/A	Clock Input
I	CLK1	CLK1	Bit	N/A	Clock Input
I	CLK2	CLK2	Bit	N/A	Clock Input
I	CLK3	CLK3	Bit	N/A	Clock Input
I	CE0	CE0	Bit	N/A	Clock Enable Input
I	CE1	CE1	Bit	N/A	Clock Enable Input
I	CE2	CE2	Bit	N/A	Clock Enable Input
I	CE3	CE3	Bit	N/A	Clock Enable Input
I	RST0	RST0	Bit	N/A	Reset Input
I	RST1	RST1	Bit	N/A	Reset Input
I	RST2	RST2	Bit	N/A	Reset Input
I	RST3	RST3	Bit	N/A	Reset Input
O	P	P[17:0]	Bus	17:0	Product
O	SROA	SROA[8:0]	Bus	8:0	Shift A Output
O	SROB	SROB[8:0]	Bus	8:0	Shift B Output
O	ROA	ROA[8:0]	Bus	8:0	Registered Output A from Multiplier
O	ROB	ROB[8:0]	Bus	8:0	Registered Output B from Multiplier
O	SIGNEDP	SIGNEDP	Bit	N/A	Output Sign Bit (result of SignedA or SignedB)

MULT9X9C Attribute Description

Table 804:

Name	Value	Default	Description
REG_INPUTA_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input A clock selection
REG_INPUTA_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input A clock enable selection

Table 804:

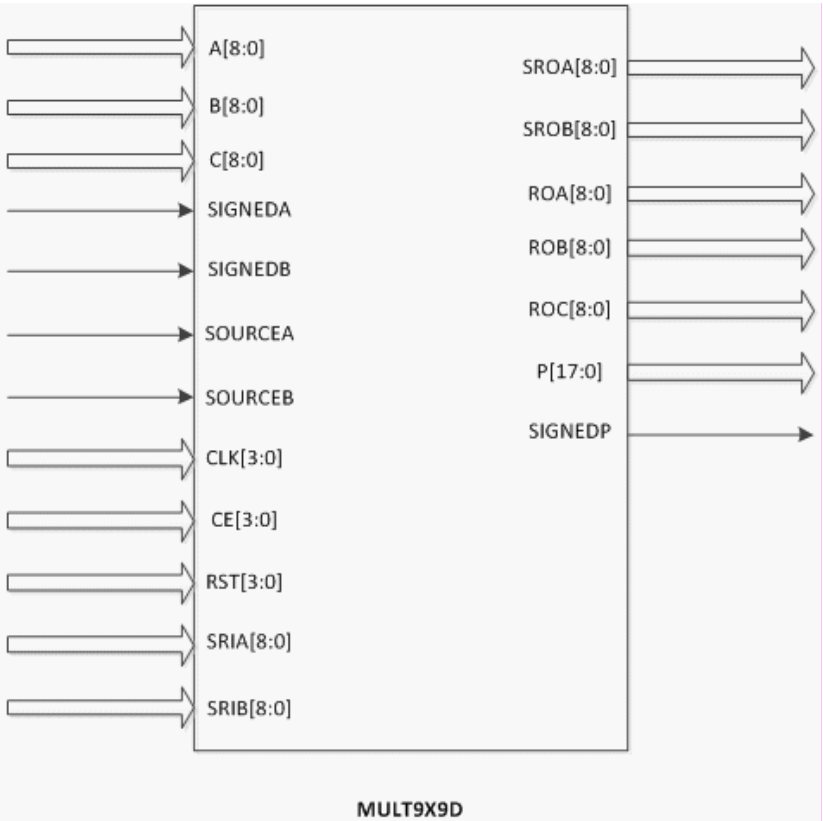
Name	Value	Default	Description
REG_INPUTA_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input A reset selection
REG_INPUTB_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input B clock selection
REG_INPUTB_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input B clock enable selection
REG_INPUTB_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input B reset selection
REG_PIPELINE_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Pipeline clock selection
REG_PIPELINE_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Pipeline clock enable selection
REG_PIPELINE_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Pipeline reset selection
REG_OUTPUT_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Output clock selection
REG_OUTPUT_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Output clock enable selection
REG_OUTPUT_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Output reset selection
CAS_MATCH_REG	"FALSE", "TRUE"	"FALSE"	Cascade match register option
MULT_BYPASS	"DISABLED", "ENABLED"	"ENABLED"	Multiplier bypass option
RESETMODE	"SYNC", "ASYNC"	"SYNC"	Global set reset selection
GSR	"ENABLED", "DISABLED"	"ENABLED"	Reset mode selection

MULT9X9D

DSP Multiplier

Architectures Supported:

- ECP5



INPUTS: A8, A7, A6, A5, A4, A3, A2, A1, A0, B8, B7, B6, B5, B4, B3, B2, B1, B0, C8, C7, C6, C5, C4, C3, C2, C1, C0, SIGNEDA, SIGNEDB, SOURCEA, SOURCEB, CE3, CE2, CE1, CE0, CLK3, CLK2, CLK1, CLK0, RST3, RST2, RST1, RST0, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, ROA8, ROA7, ROA6, ROA5, ROA4, ROA3, ROA2, ROA1, ROA0, ROB8, ROB7, ROB6, ROB5, ROB4, ROB3, ROB2, ROB1, ROB0, ROC8, ROC7, ROC6, ROC5, ROC4, ROC3, ROC2, ROC1, ROC0, P17, P16, P15, P14, P13, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0, SIGNEDP

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTC_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTC_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTC_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

CLK0_DIV: “ENABLED” (default) “DISABLED”

CLK1_DIV: “ENABLED” (default) “DISABLED”

CLK2_DIV: “ENABLED” (default) “DISABLED”

CLK3_DIV: “ENABLED” (default) “DISABLED”

CAS_MATCH_REG: "FALSE" (default), "TRUE"

MULT_BYPASS: "DISABLED" (default), "ENABLED"

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

SOURCEB_MODE"B_SHIFT" (default),"C_SHIFT","B_C_DYNAMIC","HIGHSPEED"

MULT9X9D Port Description

Table 805:

I/O	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	A	A[8:0]	Bus	8:0	A input
I	B	B[8:0]	Bus	8:0	B input
I	C	C[8:0]	Bus	8:0	C input
I	SRIA	SRIA[8:0]	Bus	8:0	Shift input A
I	SRIB	SRIB[8:0]	Bus	8:0	Shift input B
I	SIGNEDA	SIGNEDA	Bit	N/A	Input A sign selection

Table 805:

I/O	Port Name	Capture Name	Type	Size (Buses Only)	Description
I	SIGNEDB	SIGNEDB	Bit	N/A	Input B sign selection
I	SOURCEA	SOURCEA	Bit	N/A	Source A selection
I	SOURCEB	SOURCEB	Bit	N/A	Source B selection
I	CLK0	CLK0	Bit	N/A	Clock Input
I	CLK1	CLK1	Bit	N/A	Clock Input
I	CLK2	CLK2	Bit	N/A	Clock Input
I	CLK3	CLK3	Bit	N/A	Clock Input
I	CE0	CE0	Bit	N/A	Clock Enable Input
I	CE1	CE1	Bit	N/A	Clock Enable Input
I	CE2	CE2	Bit	N/A	Clock Enable Input
I	CE3	CE3	Bit	N/A	Clock Enable Input
I	RST0	RST0	Bit	N/A	Reset Input
I	RST1	RST1	Bit	N/A	Reset Input
I	RST2	RST2	Bit	N/A	Reset Input
I	RST3	RST3	Bit	N/A	Reset Input
O	P	P[17:0]	Bus	17:0	Product
O	SROA	SROA[8:0]	Bus	8:0	Shift A Output
O	SROB	SROB[8:0]	Bus	8:0	Shift B Output
O	SROC	SROC[8:0]	Bus	8:0	Shift C Output
O	ROA	ROA[8:0]	Bus	8:0	Registered Output A from Multiplier
O	ROB	ROB[8:0]	Bus	8:0	Registered Output B from Multiplier
O	SIGNEDP	SIGNEDP	Bit	N/A	Output Sign Bit (result of SignedA or SignedB)

MULT9X9D Attribute Description

Table 806:

Name	Value	Default	Description
REG_INPUTA_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input A clock selection
REG_INPUTA_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input A clock enable selection

Table 806:

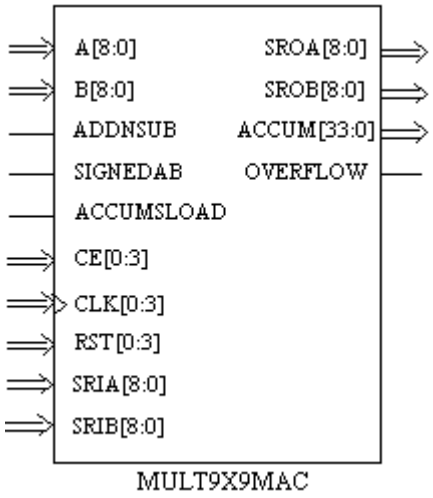
Name	Value	Default	Description
REG_INPUTA_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input A reset selection
REG_INPUTB_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input B clock selection
REG_INPUTB_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input B clock enable selection
REG_INPUTB_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input B reset selection
REG_INPUTC_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Input C clock selection
REG_INPUTC_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Input C clock enable selection
REG_INPUTC_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Input C reset selection
REG_PIPELINE_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Pipeline clock selection
REG_PIPELINE_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Pipeline clock enable selection
REG_PIPELINE_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Pipeline reset selection
REG_OUTPUT_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	"NONE"	Output clock selection
REG_OUTPUT_CE	"CE0", "CE1", "CE2", "CE3"	"CE0"	Output clock enable selection
REG_OUTPUT_RST	"RST0", "RST1", "RST2", "RST3"	"RST0"	Output reset selection
CLK0_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK0 divider setting.
CLK1_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK1 divider setting.
CLK2_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK2 divider setting.
CLK3_DIV	“ENABLED”, “DISABLED”	"ENABLED"	CLK3 divider setting.
HIGHSPEED_CLK	"NONE", "CLK0", "CLK1", "CLK2", "CLK3"	NONE"	High speed clock setting.
CAS_MATCH_REG	"FALSE", "TRUE"	"FALSE"	Cascade match register option
SOURCEB_MODE	"B_SHIFT", "C_SHIFT", "B_C_DYNAMIC", "HIGHSPEED"	"B_SHIFT"	SOURCEB mode.
MULT_BYPASS	"DISABLED", "ENABLED"	"ENABLED"	Multiplier bypass option
RESETMODE	"SYNC", "ASYNC"	"SYNC"	Global set reset selection
GSR	"ENABLED", "DISABLED"	"ENABLED"	Reset mode selection

MULT9X9MAC

ECP DSP Multiplier

Architectures Supported:

- LatticeECP (DSP Blocks Only)



INPUTS: A8, A7, A6, A5, A4, A3, A2, A1, A0, B8, B7, B6, B5, B4, B3, B2, B1, B0, ADDNSUB, SIGNEDAB, ACCUMSLOAD, CE0, CE1, CE2, CE3, CLK0, CLK1, CLK2, CLK3, RST0, RST1, RST2, RST3, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0

OUTPUTS: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, ACCUM33, ACCUM32, ACCUM31, ACCUM30, ACCUM29, ACCUM28, ACCUM27, ACCUM26, ACCUM25, ACCUM24, ACCUM23, ACCUM22, ACCUM21, ACCUM20, ACCUM19, ACCUM18, ACCUM17, ACCUM16, ACCUM15, ACCUM14, ACCUM13, ACCUM12, ACCUM11, ACCUM10, ACCUM9, ACCUM8, ACCUM7, ACCUM6, ACCUM5, ACCUM4, ACCUM3, ACCUM2, ACCUM1, ACCUM0, OVERFLOW

ATTRIBUTES:

REG_INPUTA_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTA_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTA_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_INPUTB_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_INPUTB_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_INPUTB_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_PIPELINE_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_PIPELINE_CE: "CE0" (default), "CE1", "CE1", "CE3"

REG_PIPELINE_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_OUTPUT_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_OUTPUT_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_OUTPUT_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_SIGNEDAB_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_SIGNEDAB_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_SIGNEDAB_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ACCUMSLOAD_0_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ACCUMSLOAD_0_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ACCUMSLOAD_0_RST: "RST0" (default), "RST1", "RST2", "RST3"

REG_ACCUMSLOAD_1_CLK: "NONE" (default), "CLK0", "CLK1", "CLK2", "CLK3"

REG_ACCUMSLOAD_1_CE: "CE0" (default), "CE1", "CE2", "CE3"

REG_ACCUMSLOAD_1_RST: "RST0" (default), "RST1", "RST2", "RST3"

SHIFT_IN_A: "FALSE" (default), "TRUE"

SHIFT_IN_B: "FALSE" (default), "TRUE"

GSR: "ENABLED" (default), "DISABLED"

Description

MULT9X9MAC supports operand bit widths for 9X9 multiplication and accumulates the output up to 34 bits. The DSP block includes optional registers for the input and intermediate pipeline stage. Pipeline stages may be set using a pipeline attribute. The output registers are required for the accumulator. Signed and unsigned arithmetic are supported. The

OVERFLOW bit is also provided when the accumulated results are in the overflow condition. ACCUMSLOAD determines the mode of operation for either loading the multiplier product or to accumulate.

The primitive consists of three types of optional pipeline registers:

- Input registers, located before the multipliers and registering the operands
- Multiplier pipeline registers, located after the multipliers and product registration
- Output registers, located before leaving the block, and registering the mode-specific output.

See the description of MULT18X18 for more information on control signals for DSP blocks.

Refer to the following technical note on the Lattice web site.

- TN1057 - LatticeECP sysDSP Usage Guide

MULT9X9MAC pin functions:

Table 807:

Function	Pins
input data A and B	A[8:0], B[8:0]
signed input (0 = unsigned, 1 = signed)	SIGNEDAB
add/subtract (0 = add, 1 = subtract)	ADDNSUB
Accumulate (HIGH) /Load (LOW) Mode	ACCUMSLOAD
clock enable	CE[0:3]
clock input	CLK[0:3]
reset	RST[0:3]
shifted input A and B (from previous stage)	SRIA[8:0], SRIB[8:0]
shifted output A and B (from previous stage)	SROA[8:0], SROB[8:0]
output data	ACCUM[33:0]
overflow	OVERFLOW

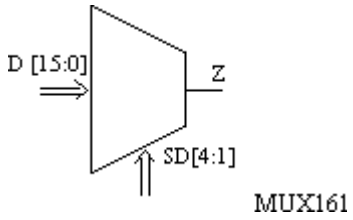
MUX161

16-Input Mux within the PFU (4 Slices)

Architectures Supported:

- ECP5
- LatticeECP/EC

- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, SD1, SD2, SD3, SD4

OUTPUT: Z

Description

For more usage, see related technical notes or contact technical support.

Truth Table

Table 808:

INPUTS	OUTPUTS	INPUTS	OUTPUTS
SD[4:1]	Z	SD[4:1]	Z
0000	D0	1000	D8
0001	D1	1001	D9
0010	D2	1010	D10
0011	D3	1011	D11
0100	D4	1100	D12
0101	D5	1101	D13
0110	D6	1110	D14
0111	D7	1111	D15

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

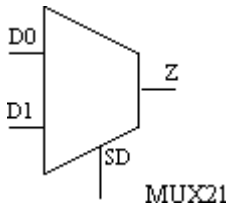
MUX21

2-to-1 Mux

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, SD

OUTPUT: Z

Description

For more usage, see related technical notes or contact technical support.

Truth Table

Table 809:

INPUTS			OUTPUTS
D0	D1	SD	Z
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1

X = Don’t care

Note

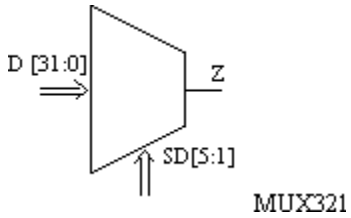
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

MUX321

32-Input Mux within the PFU (8 Slices)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21, D22, D23, D24, D25, D26, D27, D28, D29, D30, D31, SD1, SD2, SD3, SD4, SD5

OUTPUT: Z

Description

For more usage, see related technical notes or contact technical support.

Truth Table

Table 810:

INPUTS	OUTPUTS	INPUTS	OUTPUTS
SD[5:1]	Z	SD[5:1]	Z
00000	D0	10000	D16
00001	D1	10001	D17
00010	D2	10010	D18
00011	D3	10011	D19
00100	D4	10100	D20
00101	D5	10101	D21
00110	D6	10110	D22
00111	D7	10111	D23
01000	D8	11000	D24
01001	D9	11001	D25
01010	D10	11010	D26
01011	D11	11011	D27
01100	D12	11100	D28
01101	D13	11101	D29
01110	D14	11110	D30
01111	D15	11111	D31

Note

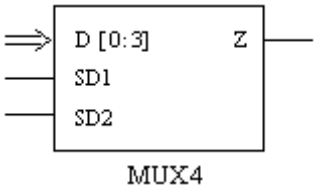
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

MUX4

4-bit Multiplexer

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: D0, D1, D2, D3, SD1, SD2

OUTPUT: Z

Description

For more usage, see related technical notes or contact technical support.

Note

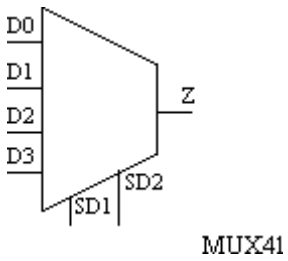
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

MUX41

4 to 1 Mux

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



MUX41

INPUTS: D0, D1, D2, D3, SD1, SD2

OUTPUT: Z

Description

For more usage, see related technical notes or contact technical support.

Truth Table

Table 811:

INPUTS						OUTPUTS
D0	D1	D2	D3	SD1	SD2	Z
0	X	X	X	0	0	0
1	X	X	X	0	0	1
X	0	X	X	1	0	0
X	1	X	X	1	0	1
X	X	0	X	0	1	0
X	X	1	X	0	1	1
X	X	X	0	1	1	0
X	X	X	1	1	1	1

X = Don’t care

Note

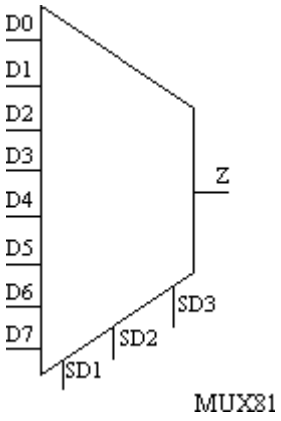
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

MUX81

8 to 1 Mux

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



MUX81

INPUTS: D0, D1, D2, D3, D4, D5, D6, D7, SD1, SD2, SD3

OUTPUT: Z

Description

For more usage, see related technical notes or contact technical support.

Truth Table

Table 812:

INPUTS									OUTPUTS
D0	D1	D2	D3	D4	D5	D6	D7	SD[1:3]	Z
0	X	X	X	X	X	X	X	0 0 0	0
1	X	X	X	X	X	X	X	0 0 0	1
X	0	X	X	X	X	X	X	1 0 0	0
X	1	X	X	X	X	X	X	1 0 0	1
X	X	0	X	X	X	X	X	0 1 0	0
X	X	1	X	X	X	X	X	0 1 0	1
X	X	X	0	X	X	X	X	1 1 0	0
X	X	X	1	X	X	X	X	1 1 0	1
X	X	X	X	0	X	X	X	0 0 1	0
X	X	X	X	1	X	X	X	0 0 1	1
X	X	X	X	X	0	X	X	1 0 1	0
X	X	X	X	X	1	X	X	1 0 1	1
X	X	X	X	X	X	0	X	0 1 1	0
X	X	X	X	X	X	1	X	0 1 1	1
X	X	X	X	X	X	X	0	1 1 1	0
X	X	X	X	X	X	X	1	1 1 1	1

X = Don’t care

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

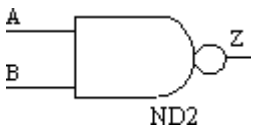
N

ND2

2 Input NAND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B

OUTPUT: Z

Note

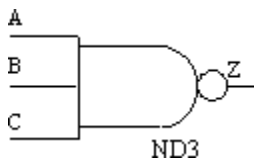
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ND3

3 Input NAND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C

OUTPUT: Z

Note

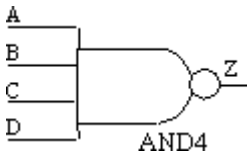
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ND4

4 Input NAND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

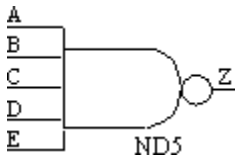
ND5

5 Input NAND Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3

- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

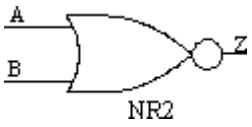
NR2

2 Input NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2

- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

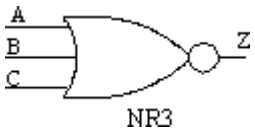
NR3

3 Input NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF

- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

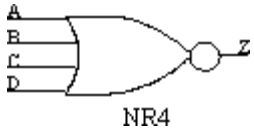
NR4

4 Input NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D

- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D

OUTPUT: Z

Note

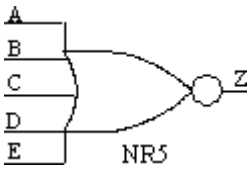
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

NR5

5 Input NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

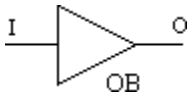
O

OB

Output Buffer

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: I

OUTPUT: O

Truth Table

Table 813:

INPUTS	OUTPUTS
I	O
1	1
0	0
Z	U

U = Unknown

Note

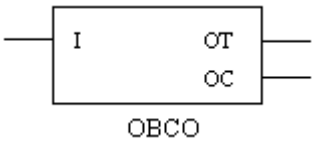
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OBCO

Output Complementary Buffer

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUT: I

OUTPUTS: OT, OC

Note

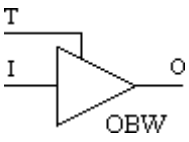
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OBW

Output Buffer with Tristate

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: I, T

OUTPUT: O

Truth Table

Table 814:

INPUTS		OUTPUTS
I	T	O
0	1	weak 0
1	1	weak 1

Note

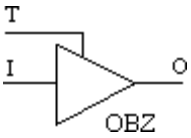
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OBZ

Output Buffer with Tristate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: I, T

OUTPUT: O

Truth Table

Table 815:

INPUTS		OUTPUTS
I	T	O
X	1	Z
0	0	0
1	0	1

X = Don’t care

When TSALL=0, O=Z

Note

- For PU/PD buffers, when TSALL=0, O will be pulled up or pulled down, respectively. The letters PU (PD) in the buffer name indicate that a pull-up (pull-down) primitive is available. This is used to generate a logic high level (low level) for nodes that may be floating.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

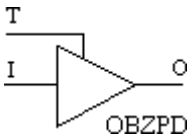
OBZPD

Output Buffer with Tristate and Pull-down

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2
- MachXO

- Platform Manager



INPUTS: I, T

OUTPUT: O

Truth Table

Table 816:

INPUTS		OUTPUTS
I	T	O
X	1	Z
0	0	0
1	0	1

X = Don’t care

When TSALL=0, O=Z

Note

- For PU/PD buffers, when TSALL=0, O will be pulled up or pulled down, respectively. The letters PU (PD) in the buffer name indicate that a pull-up (pull-down) primitive is available. This is used to generate a logic high level (low level) for nodes that may be floating.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

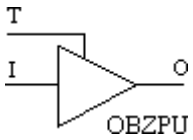
OBZPU

Output Buffer with Tristate and Pull-up

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP

- LatticeXP2
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: I, T

OUTPUT: O

Truth Table

Table 817:

INPUTS		OUTPUTS
I	T	O
X	1	Z
0	0	0
1	0	1

X = Don’t care

When TSALL=0, O=Z

Note

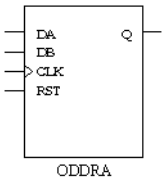
- For PU/PD buffers, when TSALL=0, O will be pulled up or pulled down, respectively. The letters PU (PD) in the buffer name indicate that a pull-up (pull-down) primitive is available. This is used to generate a logic high level (low level) for nodes that may be floating.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ODDRA

Output DDR

Architectures Supported:

- LatticeSC/M

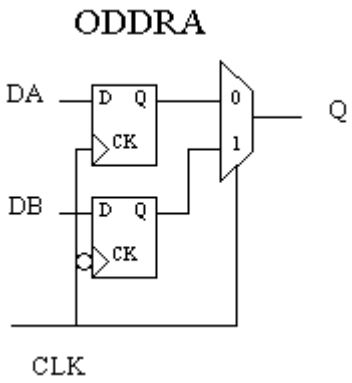


INPUTS: DA, DB, CLK, RST

OUTPUT: Q

Description

Output DDR data (positive edge and negative edge data) to the buffer. The following symbolic diagram shows the flip-flop structure of this primitive.

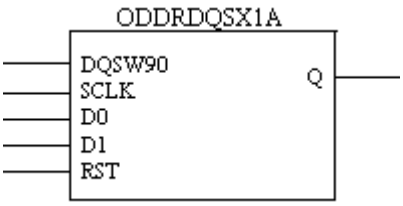


ODDRDQSX1A

Output for DDR1/2 Memory

Architectures Supported:

- MachXO2
- Platform Manager 2



INPUTS: DQSW90, SCLK, D0, D1, RST

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

ODDRDQSX1A is the output for DDR1/2 memory using the PIC hardware cell. It is used for right side only. See the below table for the port description.

Table 818:

Signal	I/O	Description
DQSW90	I	Shifts the DQS signal by 90 degree, from DQSBUFH
SCLK	I	Clock from the CIB
D0	I	Data A primary phase of data (first out)
D1	I	Data B secondary phase of data (second out)
RST	I	RESET to this block from the CIB
Q	O	DDR output data

For more information and usage, refer to the following technical note on the Lattice web site.

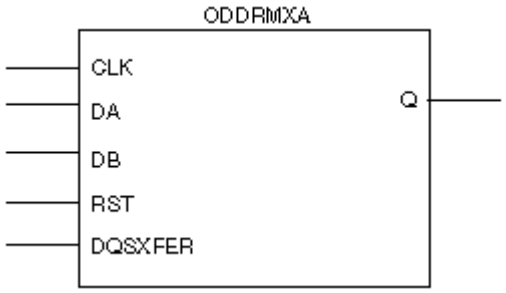
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

ODDRMXA

DDR Output Registers

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: CLK, DA, DB, RST, DQSXFER

OUTPUT: Q

Description

The ODDRMXA primitive implements the output register for both write and tristate functions. This primitive is used to output DDR data and DQS strobe to the memory. All DDR output tristate functions are also implemented using this primitive.

The following table provides description of all I/O ports associated with the ODDRMXA primitive.

Table 819:

Port Name	I/O	Definition
CLK	I	System CLK or ECLK
DA	I	Data at the negative edge of the clock
DB	I	Data at the positive edge of the clock
RST	I	Reset
DQSXFER	I	90-degree phase shifted clock coming from the DQSBUFC block
Q	O	DDR data to the memory

Notes:

- RST should be held low during DDR Write operation. By default the software will implement CE High and RST low.
- DDR output and tristate registers do not have CE support. RST is available for tristate DDRX mode (while reading). LSR will default to set when used in tristate mode.
- When asserting reset during DDR writes, it is important to know that it resets only the flip-flops but not the latches.

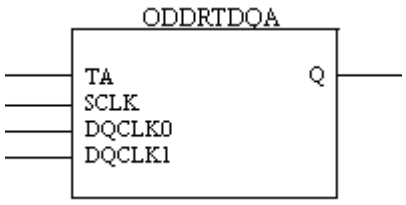
For more usage, see related technical notes or contact technical support.

ODDRTDQA

Tri-State for DQ: DDR3_MEM and DDR_GENX2

Architectures Supported:

- LatticeECP3



INPUTS: TA, SCLK, DQCLK1, DQCLK0

OUTPUT: Q

Description

ODDRTDQA is the tri-state for DQ used for DDR3_MEM (DDR3 memory mode) and DDR_GENX2.

- E and EA: DDR3_MEM and DDR_GENX2 (left/right)

See the below table for the port description.

Table 820:

Signal	I/O	Description
TA	I	Tri-state input.
SCLK	I	System clock.
DQCLK0	I	One clock edge, at the frequency of SCLK, used in output gearing, 90 degree out of phase from DQCLK1.
DQCLK1	I	One clock edge, at the frequency of SCLK, used in output gearing.
Q	O	Tri-state output.

For more information and usage, refer to the following technical note on the Lattice web site.

- TN1177 - LatticeECP3 sysIO Usage Guide

ODDRTDQSA

Tri-State for Single-Ended and Differential DQS: DDR_MEM, DDR2_MEM, and DDR3_MEM

Architectures Supported:

- LatticeECP3



INPUTS: TA, SCLK, DQSTCLK, DQSW, DB

OUTPUT: Q

Description

ODDRTDQSA is the tri-state for single-ended and differential DQS, used in DDR_MEM (DDR memory mode), DDR2_MEM (DDR2 memory mode), and DDR3_MEM (DDR3 memory mode).

- E and EA: DDR_MEM and DDR2_MEM (left/right/top)
- E and EA: DDR3_MEM (left/right)

See the below table for the port description.

Table 821:

Signal	I/O	Description
TA	I	Tri-state input
SCLK	I	System clock
DQSW	I	DQS write clock
DQSTCLK	I	DQS tri-state clock
DB	I	Data input (ONEGB)
Q	O	DQS tri-state output

For more information and usage, refer to the following technical note on the Lattice web site.

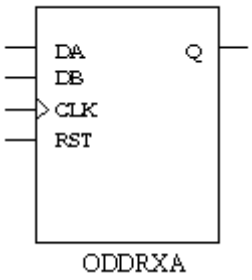
- TN1177 - LatticeECP3 sysIO Usage Guide

ODDRXA

Output DDR

Architectures Supported:

- LatticeSC/M

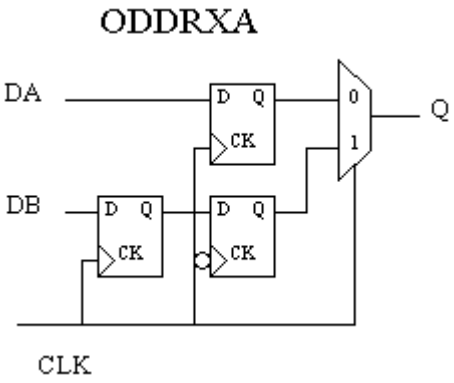


INPUTS: DA, DB, CLK, RST

OUTPUT: Q

Description

Output DDR data with half cycle clock domain transfer. The following symbolic diagram shows the flip-flop structure of this primitive.

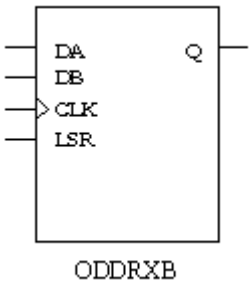


ODDRXB

Output DDR

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: DA, DB, CLK, LSR

OUTPUT: Q

ATTRIBUTES:

REGSET: "RESET" (default), "SET"

Description

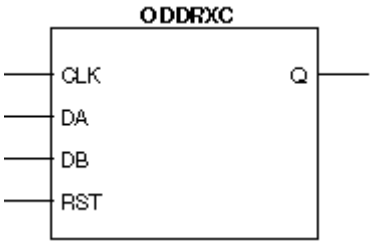
Output DDR data with half cycle clock domain transfer.

ODDRXC

DDR Generic Output

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: DA, DB, CLK, RST

OUTPUT: Q

Description

This DDR output module inputs two data streams and multiplexes them together to generate a single stream of data going to the sysIO™ buffer. CLK to this module can be connected to the edge clock or to the FPGA clock. This primitive is also used when DDR function is required for the tristate signal. See the following table for port description.

Table 822:

Port Name	I/O	Definition
DA	I	Data at the negative edge of the clock
DB	I	Data at the positive edge of the clock
CLK	I	Clock from CIB
RST	I	Reset signal
Q	O	DDR data to the memory

Notes:

- LSR should be held low during DDR Write operation. By default, the software will be implemented with CE High and LSR low.
- DDR output and tristate registers do not have CE support. LSR is available for the tristate DDRX mode (while reading). The LSR will default to set when used in the tristate mode.
- CE and LSR support is available for the regular (non-DDR) output mode.
- When asserting reset during DDR writes, it is important to keep in mind that this would only reset the flip-flops but not the latches.

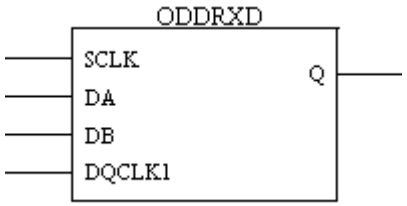
For more usage, see related technical notes or contact technical support.

ODDRXD

Output DDR for DDR_MEM, DDR2_MEM, DDR_GENX1, and DDR2_MEMGEN

Architectures Supported:

- LatticeECP3



INPUTS: SCLK, DA, DB, DQCLK1

OUTPUT: Q

ATTRIBUTES:

(*EA only*) MEMMODE: "DISABLED" (default), "ENABLED"

Description

ODDRXD is the output DDR for DDR_MEM (DDR memory mode), DDR2_MEM (DDR2 memory mode), DDR_GENX1 (DDR generic mode in X1 gearing), and DDR2_MEMGEN.

- E and EA: DDR_MEM, DDR2_MEM, and DDR2_MEMGEN (left/right/top)
- E: DDR_GENX1 (left/right/top)

The port information is described in the below table.

Table 823:

Signal	I/O	Description
SCLK	I	System clock.
DA	I	Data at the positive edge of the clock (OPOSA).
DB	I	Data at the negative edge of the clock (ONEGB).
DQCLK1	I	One clock edge, at the frequency of SCLK, used in output gearing.
Q	O	DDR data output.

For more information and usage, refer to the following technical note on the Lattice web site.

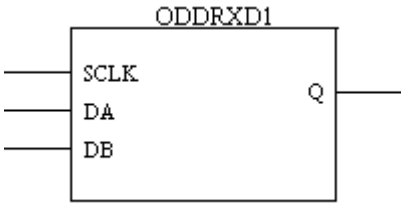
- TN1177 - LatticeECP3 sysIO Usage Guide

ODDRXD1

Output DDR for DDR_GENX1

Architectures Supported:

- LatticeECP3



INPUTS: SCLK, DA, DB

OUTPUT: Q

Description

ODDRXD1 is the output DDR for DDR_GENX1 (DDR generic mode in X1 gearing).

- EA: DDR_GENX1 (left/right/top)

The port information is described in the below table.

Table 824:

Signal	I/O	Description
SCLK	I	System clock
DA	I	Data at the positive edge of the clock (OPOSA)
DB	I	Data at the negative edge of the clock (ONEGB)
Q	O	DDR data output

For more information and usage, refer to the following technical note on the Lattice web site.

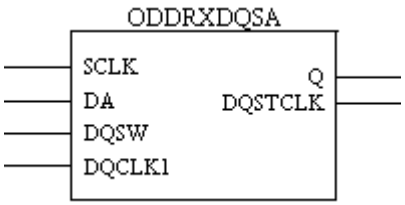
- TN1177 - LatticeECP3 sysIO Usage Guide

ODDRXDQSA

Output for Single-Ended and Differential DQS: DDR_MEM, DDR2_MEM, and DDR2_MEMGEN

Architectures Supported:

- LatticeECP3



INPUTS: SCLK, DA, DQSW, DQCLK1

OUTPUTS: Q, DQSTCLK

ATTRIBUTES:

(*EA only*) MEMMODE: "DISABLED" (default), "ENABLED"

Description

ODDRXDQSA is the output for single-ended and differential DQS, used for DDR_MEM (DDR memory mode), DDR2_MEM (DDR2 memory mode), and DDR2_MEMGEN.

- E and EA: DDR_MEM, DDR2_MEM, and DDR2_MEMGEN (left/right/top)

See the below table for the port description.

Table 825:

Signal	I/O	Description
SCLK	I	System clock.
DA	I	Data input (OPOSA).
DQSW	I	DQS write clock.
DQCLK1	I	One clock edge, at the frequency of SCLK, used in output gearing.
Q	O	DQS data output.
DQSTCLK	O	DQS Tri-state clock.

For more information and usage, refer to the following technical note on the Lattice web site.

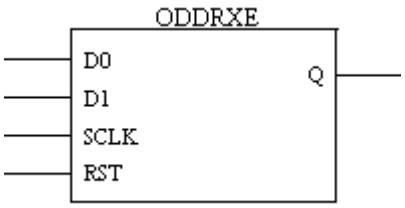
- TN1177 - LatticeECP3 sysIO Usage Guide

ODDRXE

Output for Generic DDR X1 Using 2:1 Gearing

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D0, D1, SCLK, RST

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

ODDRXE is the output for generic DDR X1 using 2:1 gearing. It uses the mPIC or PIC hardware cell. It is used for all sides.

The port information is described in the below table.

Table 826:

Signal	I/O	Description
D0	I	Data A primary phase of data (first out)
D1	I	Data B secondary phase of data (second out)
SCLK	I	Clock from the CIB
RST	I	RESET to this block from the CIB
Q	O	DDR output data

For more information and usage, refer to the following technical note on the Lattice web site.

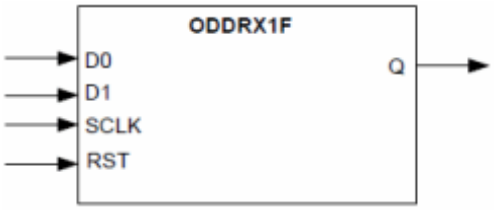
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

ODDRX1F

Generic X1 ODDR implementation

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: D0, D1, SCLK, RST

OUTPUTS: Q

Description

This primitive is used for Generic X1 ODDR implementation.

The following table gives the port description.

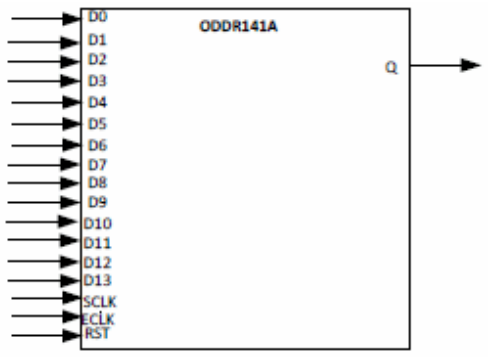
Table 827:

Signal	I/O	Description
D0	I	Data input ODDR (first to be sent out)
D1	I	Data input ODDR (second to be sent out)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	DDR data output on both edges of SCLK

ODDR141A

Architectures Supported:

- LIFMD
- LIFMDF



INPUT: D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, ECLK , SCLK, RST

OUTPUT: Q

Description

This primitive is used for Generic X1 ODDR implementation.

The following table gives the port description.

Table 828:

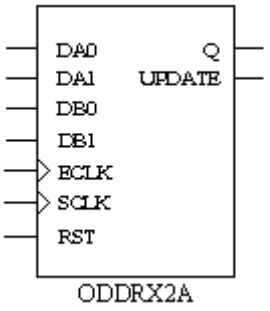
Signal	I/O	Description
D0...D13	I	Data input to the ODDR
ECLK	I	ECLK input (7x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset Input
Q	O	DDR data output on both edges of ECLK

ODDRX2A

Output DDR

Architectures Supported:

- LatticeSC/M

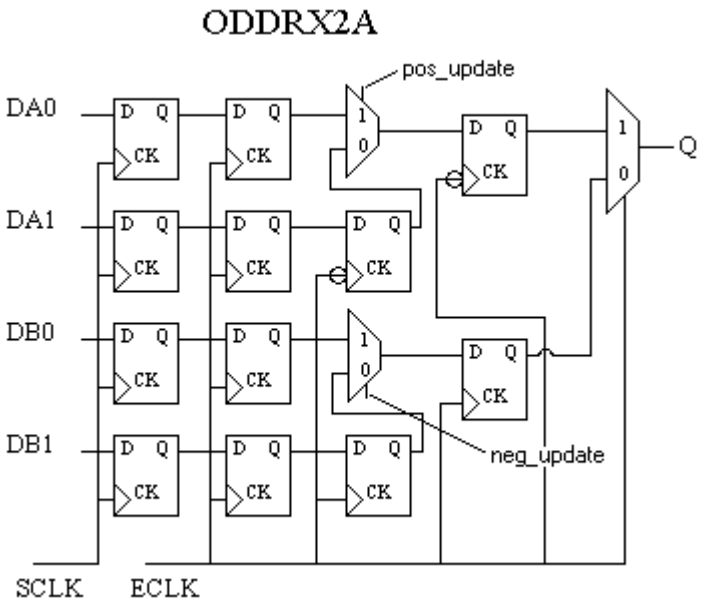


INPUTS: DA0, DA1, DB0, DB1, ECLK, SCLK, RST

OUTPUTS: Q, UPDATE

Description

Outputs DDR data to the buffer through the shift register and clock domain transfer from primary clock to edge clock. The following symbolic diagram shows the flip-flop structure of this primitive.



The pos_update and neg_update pins are select pins of the MUXes' internal

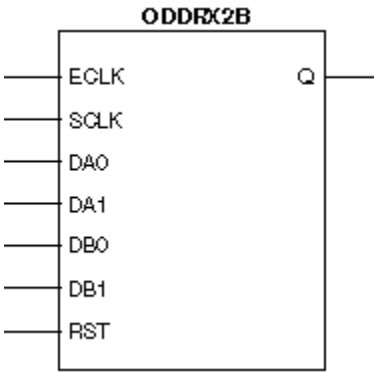
signals which go out as UPDATE signals depending upon the value of the UPDT parameter.

ODDRX2B

DDR Generic Output with 2x Gearing Ratio

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: DA0, DB0, DA1, DB1, ECLK, SCLK, RST

OUTPUT: Q

Description

This DDR output module can be used when a gearbox function is required. This primitive inputs four data streams and multiplexed them together to generate a single stream of data going to the sysIO buffer.

DDR registers of the complementary PIO is used when using this mode. The complementary PIO register can no longer be used to perform the DDR function. There are two clocks going to this primitive. ECLK is connected to the faster edge clock, while SCLK is connected to the slower FPGA clock. The DDR data output of this primitive is aligned to the faster edge clock.

Note that LSR should be held low during DDR Write operation. By default, the software will be implemented CE High and LSR low.

The following table lists port names and descriptions for the ODDRX2B primitive.

Table 829:

Port Name	I/O	Definition
DA0, DB0	I	Data at the negative edge of the clock
DA1, DB1	I	Data at the positive edge of the clock
ECLK	I	Clock connected to the faster edge clock

Table 829:

Port Name	I/O	Definition
SCLK	I	Clock connected to the slower edge clock
RST	I	Reset
Q	O	DDR data output

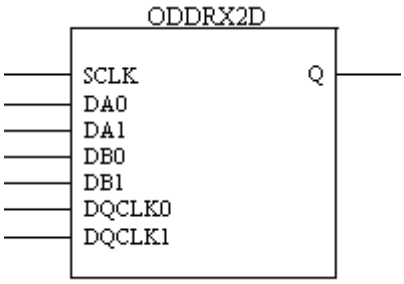
For more usage, see related technical notes or contact technical support.

ODDRX2D

Output DDR for DDR3_MEM and DDR_GENX2

Architectures Supported:

- LatticeECP3



INPUTS: SCLK, DA1, DB1, DA0, DB0, DQCLK1, DQCLK0

OUTPUT: Q

ATTRIBUTES:

ISI_CAL: "BYPASS" (default), "DEL1", "DEL2", "DEL3", "DEL4", "DEL5", "DEL6", "DEL7"

(*EA only*) MEMMODE: "DISABLED" (default), "ENABLED"

Description

ODDRX2D is the output DDR for DDR3_MEM (DDR3 memory mode) and DDR_GENX2 (DDR generic mode in X2 gearing).

- E and EA: DDR3_MEM and DDR_GENX2 (left/right)

The below table describes the port information.

Table 830:

Signal	I/O	Description
SCLK	I	System clock.
DA0	I	First data at the positive edge of the clock (OPOSA).
DA1	I	First data at the negative edge of the clock (OPOSB).
DB0	I	Second data at the positive edge of the clock (ONEGA).
DB1	I	Second data at the negative edge of the clock (ONEGB).
DQCLK0	I	One clock edge, at half the frequency of ECLK, used in output gearing, 90 degree out of phase from DQCLK1.
DQCLK1	I	One clock edge, at half the frequency of ECLK, used in output gearing.
Q	O	DDR data output.

For more information and usage, refer to the following technical note on the Lattice web site.

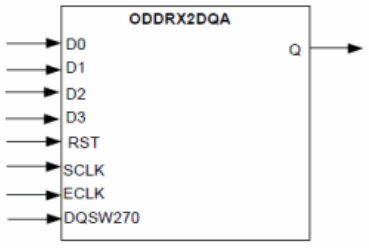
- TN1177 - LatticeECP3 sysIO Usage Guide

ODDRX2DQA

Memory Output DDR Primitive for DQ outputs

Architectures Supported:

- ECP5



INPUTS: D0, D1, D2, D3, RST, SCLK, ECLK, DQSW270

OUTPUTS: Q

Description

The following table gives the port description.

Table 831:

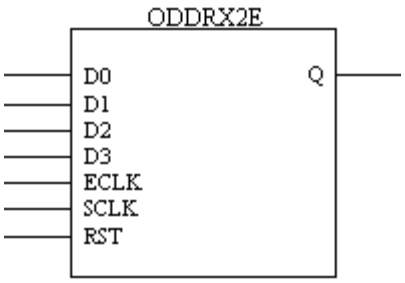
Signal	I/O	Description
D0, D1, D2, D3	I	Data input to the ODDR
RST	I	Reset input
ECLK	I	Fast Edge clock output
DQSW270	I	Clock that is 270 degrees ahead of the clock used to generate the DQS output.
SCLK	I	SCLK input
Q	O	DDR data output on both edges of DQSW270

ODDRX2E

Output for Generic DDR X2 Using 4:1 Gearing

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D0, D1, D2, D3, ECLK, SCLK, RST

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

ODDRX2E is the output for generic DDR X2 using 4:1 gearing. It uses the VPIC_TX hardware cell. It is used for top bank only. See the below table for port information.

Table 832:

Signal	I/O	Description
D0, D2	I	Data at the same edge of the clock
D1, D3	I	Data at the same edge of the clock
ECLK	I	Edge clock
SCLK	I	Clock from the CIB
RST	I	RESET to this block from the CIB
Q	O	DDR output data

For more information and usage, refer to the following technical note on the Lattice web site.

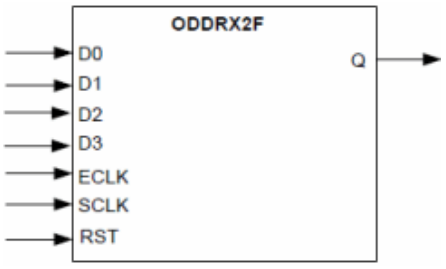
- TN1203 - Implementing High-Speed Interfaces with MachXO2 Devices

ODDRX2F

Generic X2 ODDR implementation

Architectures Supported:

- ECP5
- LIFMD
- LIFMDF



INPUTS: D0, D1, D2, D3, ECLK, SCLK, RST

OUTPUTS: Q

Description

This primitive is used for Generic X2 ODDR implementation.

The following table gives the port description.

Table 833:

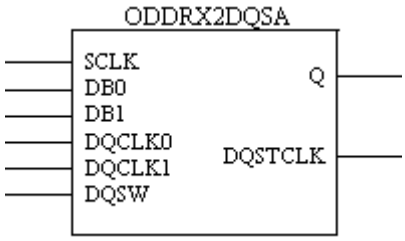
Signal	I/O	Description
D0, D2	I	Data input to the ODDR (sent out on the same edge)
D1, D3	I	Data input to the ODDR (sent out on the same edge)
ECLK	I	ECLK input (2x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	DDR data output on both edges of ECLK

ODDRX2DQSA

Output for Differential DQS: DDR3_MEM

Architectures Supported:

- LatticeECP3



INPUTS: SCLK, DB1, DB0, DQCLK1, DQCLK0, DQSW

OUTPUTS: Q, DQSTCLK

ATTRIBUTES:

ISI_CAL: "BYPASS" (default), "DEL1", "DEL2", "DEL3", "DEL4", "DEL5", "DEL6", "DEL7"

(*EA only*) MEMMODE: "DISABLED" (default), "ENABLED"

Description

ODDRX2DQSA is the output for differential DQS used for DDR3_MEM (DDR3 memory mode).

- E and EA: DDR3_MEM (left/right)

See the below table for the port description.

Table 834:

Signal	I/O	Description
SCLK	I	System clock.
DB0	I	Data input (OPOSA).
DB1	I	Data input (OPOSB).
DQSW	I	DQS write clock.
DQCLK0	I	One clock edge, at half the frequency of ECLK, used in output gearing, 90 degree out of phase from DQCLK1.
DQCLK1	I	One clock edge, at half the frequency of SCLK, used in output gearing.
Q	O	DDR data output.
DQSTCLK	O	DQS Tri-state clock.

For more information and usage, refer to the following technical note on the Lattice web site.

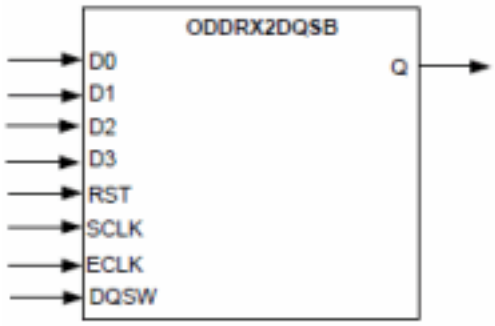
- TN1177 - LatticeECP3 sysIO Usage Guide

ODDRX2DQSB

Memory Output DDR Primitive for DQS Output

Architectures Supported:

- ECP5



INPUTS: D0, D1, D2, D3, RST, SCLK, ECLK, DQSW

OUTPUTS: Q

Description

The following table gives the port description.

Table 835:

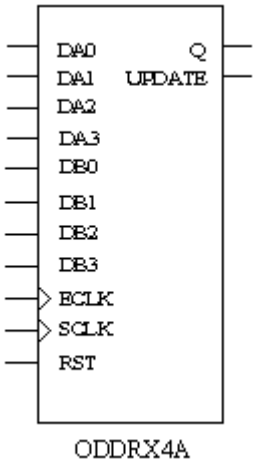
Signal	I/O	Description
D0, D1, D2, D3	I	Data input to the ODDR
RST	I	Reset input
ECLK	I	ECLK input
DQSW	I	DQSW includes write leveling phase shift from ECLK
SCLK	I	SCLK input
Q	O	DDR data output on both edges of DQSW

ODDRX4A

Output DDR

Architectures Supported:

- LatticeSC/M



INPUTS: DA0, DA1, DA2, DA3, DB0, DB1, DB2, DB3, ECLK, SCLK, RST

OUTPUTS: Q, UPDATE

ATTRIBUTES:

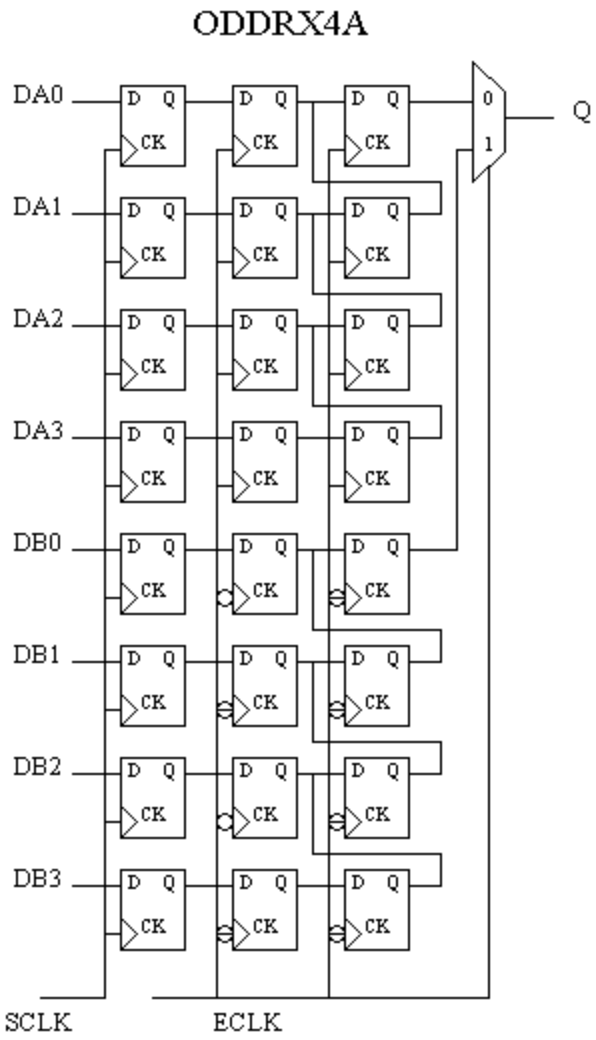
LSRMODE: "LOCAL" (default), "EDGE"

UPDT: "POS" (default), "NEG"

REGSET: "RESET" (default), "SET"

Description

Outputs DDR data to the buffer through the shift register and clock domain transfer from primary clock to edge clock. The following symbolic diagram shows the flip-flop structure of this primitive.



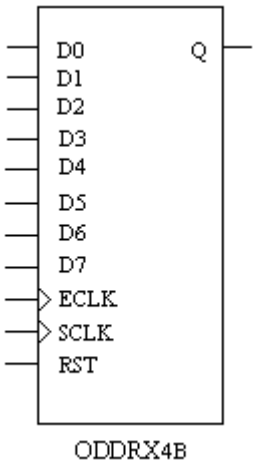
ODDRX4B

Output for Generic DDR X4 Using 8:1 Gearing

Architectures Supported:

- LFMNX

- MachXO2
- MachXO3D
- Platform Manager 2



INPUTS: D0, D1, D2, D3, D4, D5, D6, D7, ECLK, SCLK, RST

OUTPUTS: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

ODDRX4B is the output for generic DDR X4 using 8:1 gearing. It uses the VPIC_TX hardware cell. It is used for top bank only. See the below table for the port description.

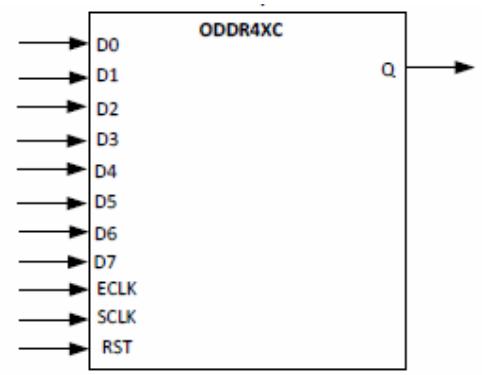
Table 836:

Signal	I/O	Description
D0, D2, D4, D6	I	Data at the same edge of the clock
D1, D3, D5, D7	I	Data at the same edge of the clock
ECLK	I	Edge clock
SCLK	I	Clock from the CIB
RST	I	RESET to this block from the CIB
Q	O	DDR output data

ODDRX4C

Architectures Supported:

- LIFMD
- LIFMDF



INPUT: D0, D1, D2, D3, D4, D5, D6, D7, ECLK , SCLK, RST

OUTPUT: Q

Description

This primitive is used for 8:1 LVDS ODDR implementation.

See the below table for the port description.

Table 837:

Signal	I/O	Description
D0...D7	I	Data input to the ODDR
ECLK	I	ECLK input (4x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset Input
Q	O	DDR data output on both edges of ECLK

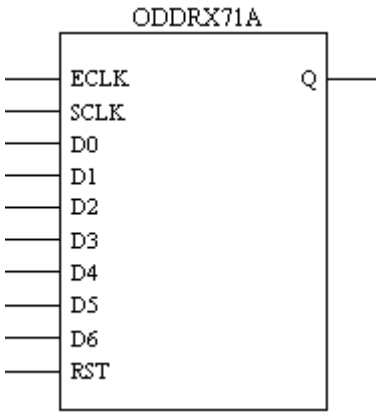
ODDRX71A

7:1 LVDS Output

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D

- MachXO3L
- Platform Manager 2



INPUTS: ECLK, SCLK, D0, D1, D2, D3, D4, D5, D6, RST

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Description

ODDRX71A is the 7:1 LVDS output that supports 7:1 gearing. It is used for top bank only. See the below table for the port description.

Table 838:

Signal	I/O	Description
ECLK	I	Edge clock
SCLK	I	Clock connected to the system clock
D0, D1, D2, D3, D4, D5, D6	I	Data available for 7:1 muxing
RST	I	RESET for this block
Q	O	7:1 LVDS signal output

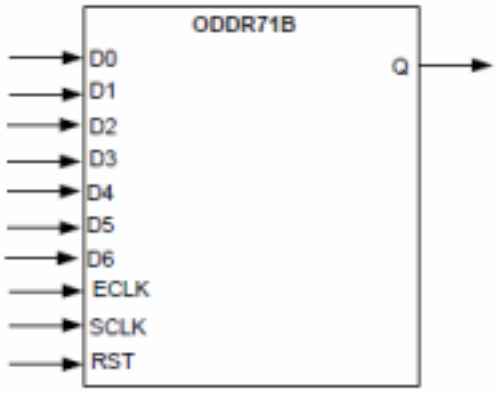
ODDR71B

7:1 LVDS ODDR implementation

Architectures Supported:

- ECP5
- LIFMD

- LIFMDF



INPUTS: D0, D1, D2, D3, D4, D5, D6, ECLK, SCLK, RST

OUTPUT: Q

Description

This primitive is used for 7:1 LVDS ODDR implementation.

Table 839:

Signal	I/O	Description
ECLK	I	ECLK input (3.5x speed of SCLK)
SCLK	I	SCLK input
D0, D1, D2, D3, D4, D5, D6	I	Data input to the ODDR
RST	I	Reset input
Q	O	DDR data output on both edges of ECLK

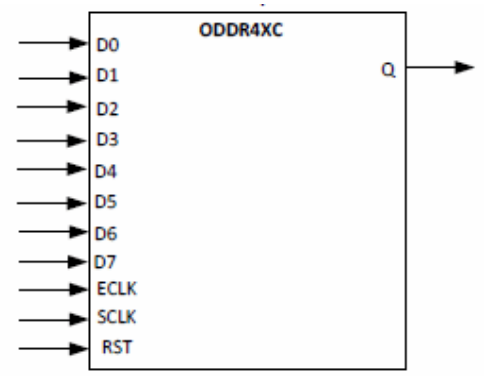
ODDRX8A

Architectures Supported:

- LIFMD
- LIFMDF

INPUT: D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15 ECLK , SCLK, RST

OUTPUT: Q



Description

This primitive is used for 16:1 LVDS ODDR implementation

See the below table for the port description.

Table 840:

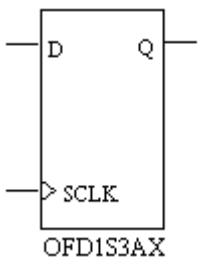
Signal	I/O	Description
D0...D15	I	Data input to the ODDR
ECLK	I	ECLK input (8x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset Input
Q	O	DDR data output on both edges of ECLK

OFD1S3AX

Positive Edge Triggered D Flip-Flop, GSR Used for Clear. Used to Tri-State DDR/DDR2

Architectures Supported:

- LatticeECP3



INPUTS: D, SCLK

OUTPUT: Q

ATTRIBUTES:

GSR: "DISABLED" (default), "ENABLED"

Description

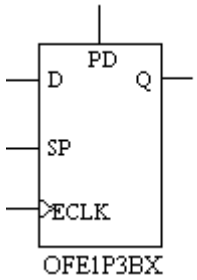
OFD1S3AX is a primitive used to implement DDR and DDR2 DQ tri-state. This primitive is functionally equivalent to the FD1S3AX primitive.

OFE1P3BX

Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and Edge Clock (used in output PIC area only)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2



INPUTS: D, SP, ECLK, PD

OUTPUT: Q

Note

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 841:

INPUTS				OUTPUTS
D	SP	ECLK	PD	Q
X	0	X	0	Q
X	X	X	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

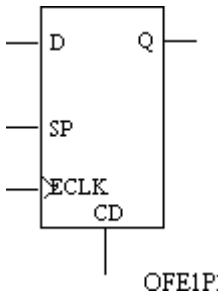
When GSR=0, Q=1 (D=SP=ECLK=PD=X)

OFE1P3DX

Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and Edge Clock (used in output PIC area only)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2



INPUTS: D, SP, ECLK, CD

OUTPUT: Q

Note

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 842:

INPUTS				OUTPUTS
D	SP	ECLK	CD	Q
X	0	X	0	Q
X	X	X	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

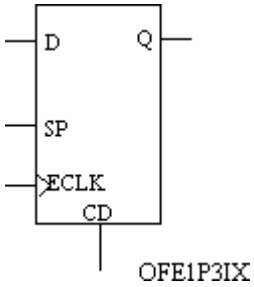
When GSR=0, Q=0 (D=SP=ECLK=CD=X)

OFE1P3IX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and Edge Clock (used in output PIC area only)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2



INPUTS: D, SP, ECLK, CD

OUTPUT: Q

Note:

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 843:

INPUTS				OUTPUTS
D	SP	ECLK	CD	Q
X	0	X	0	Q
X	X	↑	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=SP=ECLK=CD=X)

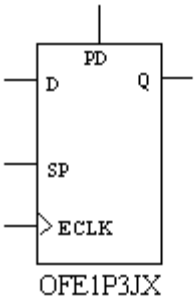
OFE1P3JX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and Edge Clock (used in output PIC area only)

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M

- LatticeSC/M
- LatticeXP
- LatticeXP2



INPUTS: D, SP, ECLK, PD

OUTPUT: Q

Note

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 844:

INPUTS				OUTPUTS
D	SP	ECLK	PD	Q
X	0	X	0	Q
X	X	↑	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

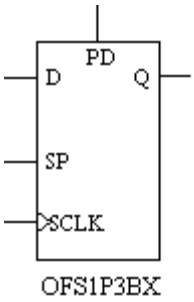
When GSR=0, Q=1 (D=SP=ECLK=PD=X)

OFS1P3BX

Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Preset, and System Clock (used in output PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SP, SCLK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 845:

INPUTS				OUTPUTS
D	SP	SCLK	PD	Q
X	0	X	0	Q
X	X	X	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=1 (D=SP=SCLK=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

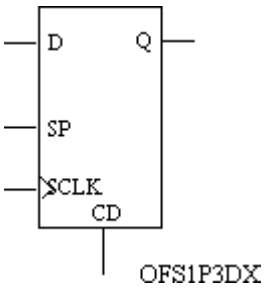
OFS1P3DX

Positive Edge Triggered D Flip-Flop with Positive Level Enable, Positive Level Asynchronous Clear, and System Clock (used in output PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2

- LIFMD
- LIFMDF
- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SP, SCLK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 846:

INPUTS				OUTPUTS
D	SP	SCLK	CD	Q
X	0	X	0	Q
X	X	X	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=SP=SCLK=CD=X)

Note

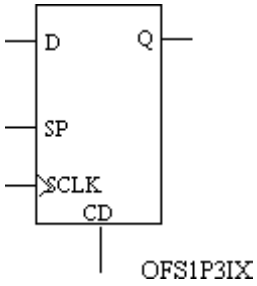
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OFS1P3IX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Clear, Positive Level Enable (Clear overrides Enable), and System Clock (used in output PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SP, SCLK, CD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 847:

INPUTS				OUTPUTS
D	SP	SCLK	CD	Q
X	0	X	0	Q
X	X	↑	1	0
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=0 (D=SP=SCLK=CD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

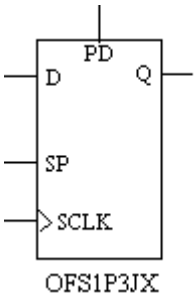
OFS1P3JX

Positive Edge Triggered D Flip-Flop with Positive Level Synchronous Preset, Positive Level Enable (Preset overrides Enable), and System Clock (used in output PIC area only)

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP

- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, SP, SCLK, PD

OUTPUT: Q

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

Note

This primitive must be paired with an output or bidirectional buffer. The mapper automatically assigns the primitive and its buffer to the same PIC. Use the PIN or LOC properties, or the LOCATE COMP preference, on the buffer per normal use, but not on this primitive.

Truth Table

Table 848:

INPUTS				OUTPUTS
D	SP	SCLK	PD	Q
X	0	X	0	Q
X	X	↑	1	1
0	1	↑	0	0
1	1	↑	0	1

X = Don’t care

When GSR=0, Q=1 (D=SP=SCLK=PD=X)

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

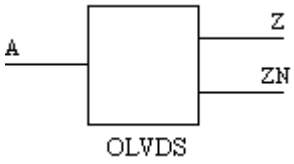
OLVDS

LVDS Output Buffer

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager

- Platform Manager 2



INPUT: A

OUTPUTS: Z, ZN

Truth Table

Table 849:

INPUTS	OUPUTS	
A	Z	ZN
0	0	1
1	1	0

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

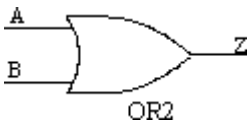
OR2

2 Input OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD

- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

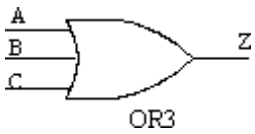
OR3

3 Input OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D

- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C

OUTPUT: Z

Note

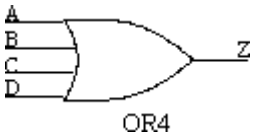
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OR4

4 Input OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D

OUTPUT: Z

Note

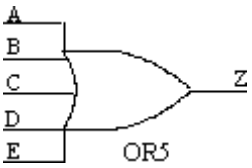
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OR5

5 Input OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E

OUTPUT: Z

Note

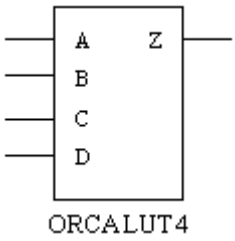
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ORCALUT4

4-Input Look Up Table

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: A, B, C, D

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 16'h0000)

Description

ORCALUT4 defines the programmed state of a LUT4 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT4 programming. The contents of the look up table are addressed by the 4 input pins to access 1 of 16 locations.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

The programming of the ORCALUT4 (that is, the 0 or 1 value of each memory location within the LUT4) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For example, hex value BF80 produces these 16 memory locations and values:

1011 1111 1000 0000

Memory location 0 (D=0, C=0, B=0, A=0) contains a 0, memory location 2 (D=0, C=0, B=1, A=0) contains a 0. Memory location 15 (D=1, C=1, B=1, A=1) contains a 1, etc.

The ORCALUT4 may encode the Boolean logic for any Boolean expression of 4 input variables. For example, if the required expression was:

Z = (D*C) + (B*!A)

then the INIT value can be derived from the truth table resulting from the expression:

```
D C B A : Z
0 0 0 0 : 0
0 0 0 1 : 0
0 0 1 0 : 1
0 0 1 1 : 0

0 1 0 0 : 0
0 1 0 1 : 0
0 1 1 0 : 1
0 1 1 1 : 0

1 0 0 0 : 0
1 0 0 1 : 0
1 0 1 0 : 1
1 0 1 1 : 0

1 1 0 0 : 1
1 1 0 1 : 1
1 1 1 0 : 1
1 1 1 1 : 1
```

INIT = F444 (16)

Adding INIT to HDL

INIT can be used as an HDL attribute. The following examples demonstrate how to use INIT with the ORCALUT4 primitive in your Verilog or VHDL source. INIT takes binary value in HDL.

Verilog Example

```
// synopsys translate_off
// parameter definition
defparam I1.init = 16'hF444 ;
// synopsys translate_on

// ORCALUT4 module instantiation
ORCALUT4 I1 (.A(A), .B (B), .C(C), .D(D), .Z(Q[0]))
/* synthesis init = "16'hF444" */;
```

VHDL Example

```
-- component definition
component ORCALUT4
   port (
         A,B,C,D: In std_logic;
               Z: Out std_logic
         );
end component;


-- attribute definition
  attribute INIT: string;
  attribute INIT of I1: label is "1000000000000000";--Z=A*B*C*D
...
-- ORCALUT4 component instantiation
I1 : ORCALUT4
   Port Map ( A=>A, B=>B, C=>C, D=>D, Z=>N_1 );
```

For generic examples of how to use HDL attributes, see “Adding FPGA Attributes to HDL” in online Help.

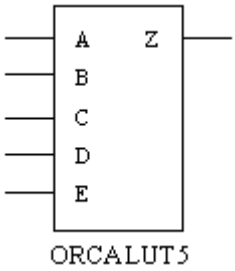
ORCALUT5

5-Input Look Up Table

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeXP
- LatticeXP2

- MachXO
- Platform Manager



INPUTS: A, B, C, D, E

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 32'h0000_0000)

Description

ORCALUT5 defines the programmed state of a LUT5 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT5 programming. The contents of the look up table are addressed by the 5 input pins to access 1 of 32 locations.

The programming of the ORCALUT5 (that is, the 0 or 1 value of each memory location within the LUT5) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For more information on INIT attribute usage, see the ORCALUT4 topic.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

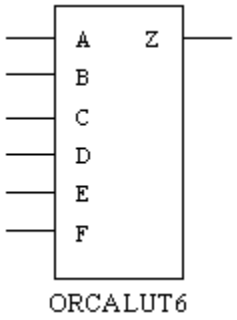
ORCALUT6

6-Input Look Up Table

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeXP

- LatticeXP2
- MachXO
- Platform Manager



INPUTS: A, B, C, D, E, F

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 64'h0000_0000_0000_0000)

Description

ORCALUT6 defines the programmed state of a LUT6 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT6 programming. The contents of the look up table are addressed by the 6 input pins to access 1 of 64 locations.

The programming of the ORCALUT6 (that is, the 0 or 1 value of each memory location within the LUT6) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For more information on INIT attribute usage, see the ORCALUT4 topic.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

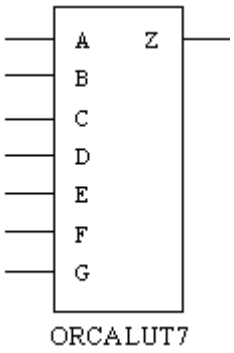
ORCALUT7

7-Input Look Up Table

Architectures Supported:

- LatticeECP/EC

- LatticeECP2/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: A, B, C, D, E, F, G

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 128'h0000_0000_0000_0000_0000_0000_0000_0000)

Description

ORCALUT7 defines the programmed state of a LUT7 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT7 programming. The contents of the look up table are addressed by the 7 input pins to access 1 of 128 locations.

The programming of the ORCALUT7 (that is, the 0 or 1 value of each memory location within the LUT7) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For more information on INIT attribute usage, see the ORCALUT4 topic.

Note

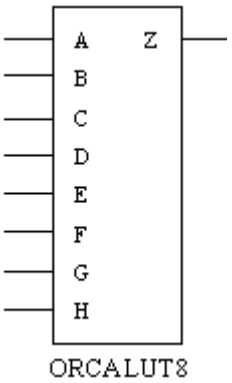
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ORCALUT8

8-Input Look Up Table

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: A, B, C, D, E, F, G, H

OUTPUT: Z

ATTRIBUTES:

INIT: hexadecimal value (default: 256'h0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000_0000)

Description

ORCALUT8 defines the programmed state of a LUT8 primitive of a Slice. While this primitive is typically targeted by logic synthesis tools, it can also be instantiated in HDL source for intimate control over LUT8 programming. The contents of the look up table are addressed by the 8 input pins to access 1 of 256 locations.

The programming of the ORCALUT8 (that is, the 0 or 1 value of each memory location within the LUT8) is determined by the value assigned with INIT. The value is expressed in hexadecimal code. Highest memory locations are in the most significant hex digit, the lowest in the least significant digit.

For more information on INIT attribute usage, see the ORCALUT4 topic.

Note

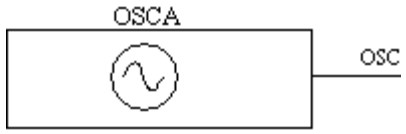
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OSCA

Internal Oscillator

Architectures Supported:

- LatticeSC/M



OUTPUT: OSC

ATTRIBUTES:

DIV: 1 (default), 2, 4, 8, 16, 32, 64, 128

Description

The OSCA is the source of the internal clock for configuration. After configuration this oscillator is disabled by default. If needed, it can be enabled by instantiating the OSCA or the Sysbus (with sys_clk_sel=OSC option). OSCA may be used as a general-purpose clock to drive FPGA logic.

The internal clock frequency can be one of eight values: 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, or 1/128 of the oscillator frequency (about 128 MHz). During start-up, the clock divider is set to 1/128 (about 1 MHz). During initialization, it is set to 1/8 (about 16 MHz). After initialization, if OSCA is enabled, it is set to the user-specified value.

Note

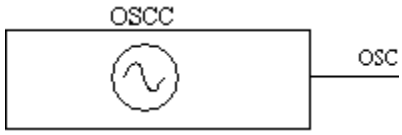
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OSCC

Internal Oscillator

Architectures Supported:

- MachXO
- Platform Manager



OUTPUT: OSC

Description

OSCC is a dedicated oscillator in the MachXO and Platform Manager device and the source of the internal clock for configuration. The oscillator frequency range is 18 to 26 MHz. The output of the oscillator can also be routed as an input clock to the clock tree. The oscillator frequency output can be further divided by internal logic (user logic) for lower frequencies, if desired. The oscillator is powered down when not in use. The example below illustrates proper usage for instantiating the OSCC primitive in VHDL.

```
COMPONENT OSCC
   PORT (OSC:OUT std_logic);
END COMPONENT;


begin


OSCInst0: OSCC
   PORT MAP (
            OSC => osc_int
         );
```

Note

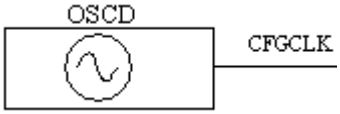
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OSCD

Oscillator for Configuration Clock

Architectures Supported:

- LatticeECP2/M



OUTPUT: CFGCLK

ATTRIBUTES:

NOM_FREQ: 2.5 (default), 4.3, 5.4, 6.9, 8.1, 9.2, 10.0, 13.0, 15.0, 20.0, 26.0, 30.0, 34.0, 41.0, 45.0, 55.0, 60.0, 130.0 (in MHz)

Description

OSCD is the primitive name of the ECP2/M oscillator. The internal oscillator is the source of the internal clock for configuration and is nominally running at 130 MHz. The oscillator is configurable by the user.

During configuration, the internal clock frequency is selected with bits 5-0 in configuration control register 0. The default frequency is 2.5 MHz, where the default values of bits 5-0 are all zeros. IO description and attribute descriptions are shown in the following tables.

OSCD Port Definition

Table 850:

Port Name	I/O	Description
CFGCLK	Output	Oscillator clock output

OSCD Usage with VHDL

```
COMPONENT OSCD
-- synthesis translate_off
   GENERIC(NOM_FREQ: string:= "2.5");
-- synthesis translate_on
   PORT (CFGCLK: OUT std_logic);
END COMPONENT;


   attribute NOM_FREQ : string;
   attribute NOM_FREQ of OSCins0 : label is "2.5";


begin


OSCInst0: OSCD
-- synthesis translate_off
   GENERIC MAP (NOM_FREQ => "2.5")
-- synthesis translate_on
   PORT MAP ( CFGCLK => osc_int);
```

OSCD Usage with Verilog HDL

```
module OSC_TOP(OSC_CLK);
output OSC_CLK;
OSCD OSCinst0 (.CFGCLK(OSC_CLK));
defparam OSCinst0.NOM_FREQ = "2.5";
```

```
endmodule
```

Note

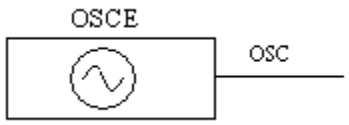
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OSCE

Oscillator for Configuration Clock

Architectures Supported:

- LatticeXP2



OUTPUT: OSC

ATTRIBUTES:

NOM_FREQ: 2.5 (default), 3.1, 4.3, 5.4, 6.9, 8.1, 9.2, 10.0, 13.0, 15.0, 20.0, 26.0, 32.0, 40.0, 54.0, 80.0, 163.0 (in MHz)

Description

OSCE is the primitive name of the XP2 oscillator. The internal oscillator is the source of the internal clock and is configurable by the user.

During configuration, the internal clock frequency is selected with bits 5-0 in configuration control register 0. The default frequency is 2.5 MHz, where the default values of bits 5-0 are all zeros. IO description and attribute descriptions are shown in the following tables.

OSCE Port Definition

Table 851:

Port Name	I/O	Description
OSC	Output	Oscillator clock output

OSCE Usage with VHDL

```
COMPONENT OSCE
-- synthesis translate_off
   GENERIC (NOM_FREQ: string := "2.5");
-- synthesis translate_on
   PORT (OSC :OUT std_logic);
```

```
END COMPONENT;


   attribute NOM_FREQ : string;
   attribute NOM_FREQ of OSCinst0 : label is "2.5";


begin


OSCInst0: OSCE
-- synthesis translate_off
   GENERIC MAP (
                NOM_FREQ => "2.5"
               )
-- synthesis translate_on
   PORT MAP (
             OSC => osc_int
             );
```

Note

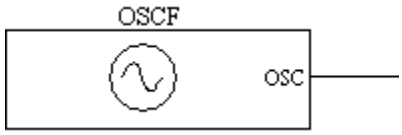
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

OSCF

Oscillator for Configuration Clock

Architectures Supported:

- LatticeECP3



OUTPUT: OSC

ATTRIBUTES:

NOM_FREQ: 2.5 (default), 4.3, 5.4, 6.9, 8.1, 9.2, 10.0, 13.0, 15.0, 20.0, 26.0, 30.0, 34.0, 41.0, 45.0, 55.0, 60.0, 130.0 (in MHz)

Description

OSCF is the primitive name of the LatticeECP3 oscillator. The internal oscillator is the source of the internal clock and is configurable by the user.

During configuration, the internal clock frequency is selected with bits 5-0 in configuration control register 0. The default frequency is 2.5 MHz, where the

default values of bits 5-0 are all zeros. IO description and attribute descriptions are shown in the following tables.

OSCF Port Definition

Table 852:

Port Name	I/O	Description
OSC	Output	Oscillator clock output

OSCF Usage with VHDL

```
COMPONENT OSCF
-- synthesis translate_off
   GENERIC  (NOM_FREQ: string :=  "2.5");
-- synthesis translate_on
   PORT (OSC: OUT std_logic);
END COMPONENT;


   attribute NOM_FREQ : string;
   attribute NOM_FREQ of OSCinst0 : label is "2.5";


begin

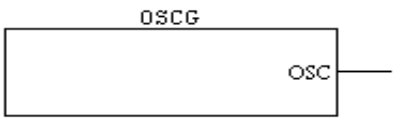

OSCInst0: OSCF
-- synthesis translate_off
   GENERIC MAP (
                NOM_FREQ => "2.5"
               )
-- synthesis translate_on
   PORT MAP (
             OSC => osc_int
            );
```

OSCG

Oscillator for ECP5

Architectures Supported:

- ECP5



OUTPUT: OSC

ATTRIBUTES:

DIV: See table below for OSCG user clock frequency values. The default Divide Ratio value is 128. Base frequency is 310 MHz.

Table 853: User Clock Frequency Values

Divide Ratio	Frequency (Ftyp) (MHz)	Divide Ratio	Frequency (Ftyp) (MHz)	Divide Ratio	Frequency (Ftyp) (MHz)	Divide Ratio	Frequency (Ftyp) (MHz)
2	155.0	18	17.2	36	8.6	72	4.3
3	103.3	19	16.3	38	8.2	76	4.1
4	77.5	20	15.5	40	7.8	80	3.9
5	62.0	21	14.8	42	7.4	84	3.7
6	51.7	22	14.1	44	7.0	88	3.5
7	44.3	23	13.5	46	6.7	92	3.4
8	38.8	24	12.9	48	6.5	96	3.2
9	34.4	25	12.4	50	6.2	100	3.1
10	31.0	26	11.9	52	6.0	104	3.0
11	28.2	27	11.5	54	5.7	108	2.9
12	25.8	28	11.1	56	5.5	112	2.8
13	23.8	29	10.7	58	5.3	116	2.7
14	22.1	30	10.3	60	5.2	120	2.6
15	20.7	31	10.0	62	5.0	124	2.5
16	19.4	32	9.7	64	4.8	128	2.4
17	18.2	34	9.1	68	4.6		

See the I/O port description in the below table.

Table 854:

Port Name	I/O	Description
OSC	Output	Oscillator clock output

Description

The OSCG element performs multiple functions on the ECP5 device. It is used for configuration, soft error detect (SED), as well as optionally in user mode. In user mode, the OSCG element has the following features:

- It permits a design to be fully self-clocked, as long as the quality of the OSCG element's silicon-based oscillator is adequate, and provides performance superior to a "roll-your-own" user-created implementation.
- If it's unused it can be turned off for power savings.

- It has a direct connection to primary clock routing through the left mid-mux.
- It can be configured for operation at a wide range of frequencies via configuration bits.
- The bandgap controller can't be shut off in ECP5 so there is no need for any caveats for BGOFF.

The one feature available to the user is the user-mode OSCG oscillator frunction.

The OSCG provides an internal clock source to user designs.

OSCG Usage with VHDL

```
COMPONENT OSCG
-- synthesis translate_off
  GENERIC (DIV: integer := 128);
-- synthesis translate_on
PORT (
  OSC      : OUT std_logic);
END COMPONENT;

attribute DIV : integer;
attribute DIV of OSCinst0 : label is 128;

begin
OSCInst0: OSCG
-- synthesis translate_off
  GENERIC MAP (DIV  => 128)
-- synthesis translate_on
PORT MAP (OSC      => OSC);
```

OSCG Usage with Verilog

```
module OSC_TOP(OSC_CLK);

output OSC_CLK;

OSCG OSCinst0 (.OSC(OSC_CLK));
defparam OSCinst0.DIV = 2;

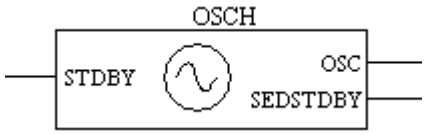
endmodule
```

OSCH

Oscillator for MachXO2/Platform Manager 2

Architectures Supported:

- MachXO2
- MachXO3L
- Platform Manager 2



INPUT: STDBY

OUTPUT: OSC, SEDSTDBY

ATTRIBUTES:

NOM_FREQ: 2.08 (default), 2.15, 2.22, 2.29, 2.38, 2.46, 2.56, 2.66, 2.77, 2.89, 3.02, 3.17, …, 19.0, 20.46, 22.17, 24.18, 26.6, 29.56, 33.25, 38.0, 44.33, 53.2, 66.5, 88.67, 133.0 (in MHz)

Description

OSCH is the primitive name of the MachXO2/Platform Manager 2 oscillator. See the IO port description in the below table.

Table 855:

Port Name	I/O	Description
STDBY	Input	Standby – power down oscillator
OSC	Output	Oscillator clock output
SEDSTDBY	Output	Standby – power down SED clock

By default, the internal oscillator will be enabled even if the user does not have it instantiated in the design. User can disable the internal oscillator by instantiating it in the design and using the STDBY port. This port can be connected to a user signal or an I/O pin. The user must insure that the oscillator is not turned off when it is needed for operations such as WISHBONE bus operations, SPI or I2C configuration, SPI or I2C user mode operation, SPI or I2C background Flash updates or SED.

OSCH Usage with VHDL

```
COMPONENT OSCH
-- synthesis translate_off
   GENERIC  (NOM_FREQ: string :=  "2.56");
-- synthesis translate_on
   PORT (STDBY   : IN  std_logic;
         OSC     : OUT std_logic;
         SEDSTDBY: OUT std_logic);
END COMPONENT;


   attribute NOM_FREQ : string;
   attribute NOM_FREQ of OSCinst0 : label is "2.56";
```

```
begin


OSCInst0: OSCH
-- synthesis translate_off
   GENERIC MAP (
                NOM_FREQ => "2.5"
               )
-- synthesis translate_on
   PORT MAP (STDBY => stdby,
             OSC   => osc_int,
             SEDSTDBY => stdby_sed
            );
```

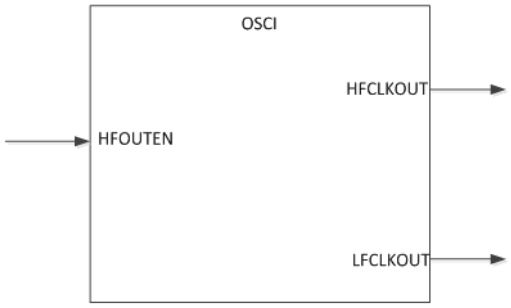
For more information, refer to the following technical note on the Lattice web site:

- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide

OSCI

Architectures Supported:

- LIFMD
- LIFMDF



INPUTS: HFOUTEN

OUTPUTS: HFCLKOUT, LFCLKOUT

See the I/O port description in the below table.

Table 856:

Port Name	I/O	Description
HFOUTEN	Input	High frequency clock output enable.

Table 856:

Port Name	I/O	Description
HFCLKOUT	Output	High frequency clock output.
LFCLKOUT	Output	Low Frequency clock output

Description

OSCI runs at 10 KHz in low frequency mode and at maximum 48 MHz in high frequency mode with output divider by 1, 2, 4 or 8.

OSCI provides internal clock sources to user designs. These clocks can directly route to the global clock network or to local fabric.

OSCI Usage with VHDL

```
Component Instantiation
Library lattice;
use lattice.components.all;

Component and Attribute Declaration
component OSCI
Generic (HFCLKDIV : Integer);
Port (
    HFOUTEN : in STD_LOGIC;
    HFCLKOUT : out STD_LOGIC;
    LFCLKOUT : out STD_LOGIC);
end component;
attribute HFCLKDIV : Integer;
attribute HFCLKDIV of I1 : label is 1; -- 1,2,4,8

OSCI Instantiation
I1: OSCI
generic map (HFCLKDIV => 1)
port map
    HFOUTEN => HFOUTEN,
    HFCLKOUT => HFCLKOUT,
    LFCLKOUT => LFCLKOUT);
```

OSCI Usage with Verilog

```
Component and Attribute Declaration
module OSCI (HFOUTEN, HFCLKOUT,
LFCLKOUT);
parameter HFCLKDIV = 1;
input HFOUTEN;
input HFCLKOUT;
output LFCLKOUT;
endmodule
OSCI Instantiation
defparam I1.HFCLKDIV = 1; // 1,2,4,8
OSCI I1 (
.HFOUTEN(HFOUTEN),
.HFCLKOUT(HFCLKOUT),
.LFCLKOUT(LFCLKOUT));
```

For more information, refer to the following technical note on the Lattice web site:

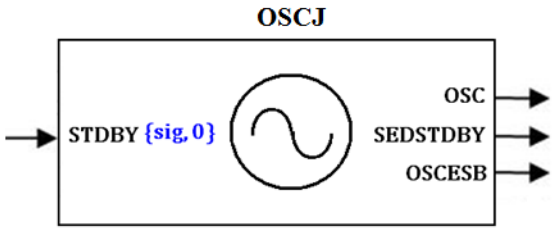
- FPGA-TN- 02015 - CrossLink sysCLOCK PLL/DLL Design and Usage Guide

OSCJ

Oscillator

Architectures Supported:

- LFMNX
- MachXO3D



INPUT: STDBY

OUTPUT: OSC, SEDSTDBY, OSCESB

ATTRIBUTES:

NOM_FREQ = 2.08 (default), 2.15, 2.22, 2.29, 2.38, 2.46, 2.56, 2.66, 2.77, 2.89, 3.02, 3.17, …, 19.0, 20.46, 22.17, 24.18, 26.6, 29.56, 33.25, 38.0,44.33, 53.2, 66.5, 88.67, 133.0 (in MHz)

Description

OSCJ has an additional output pin named OSCESB, which is connected to the input clock of ESB.

See the IO port description in the below table.

Table 857:

Port Name	I/O	Description
STDBY	I	Standby – power down oscillator
OSC	O	Oscillator clock output

Table 857:

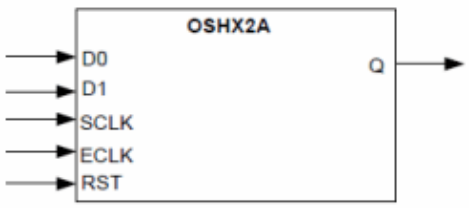
Port Name	I/O	Description
SEDSTDBY	O	Standby – power down SED clock
OSCESB	O	Oscillator clock output connected to input clock of ESB primitive The frequency is fixed at 66.5MHz

OSHX2A

Memory Output DDR Primitive for Address and Command

Architectures Supported:

- ECP5



INPUT: DO, D1, SCLK, ECLK, RST

OUTPUT: Q

Description

This primitive is used to generate the CS_N output for DDR2, DDR3, DDR3L & LPDDR memory interface.

The I/O port description are given in the following table.

Table 858:

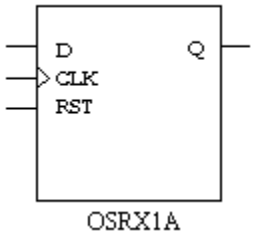
Port Name	I/O	Description
DO, D1	I	Data input
ECLK	I	ECLK input (2x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	Address and command input

OSRX1A

Output 1-Bit Shift Register

Architectures Supported:

- LatticeSC/M



INPUTS: D, CLK, RST

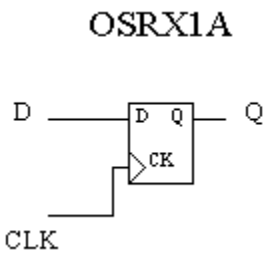
OUTPUT: Q

ATTRIBUTES:

REGSET: "RESET" (default), "SET"

Description

Outputs data through the shift register to the output data. The following symbolic diagram shows the flip-flop structure of this primitive.

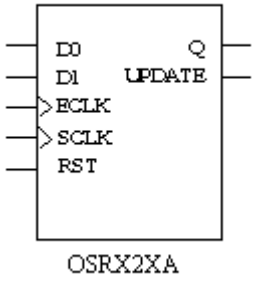


OSRX2A

Output 2-Bit Shift Register

Architectures Supported:

- LatticeSC/M

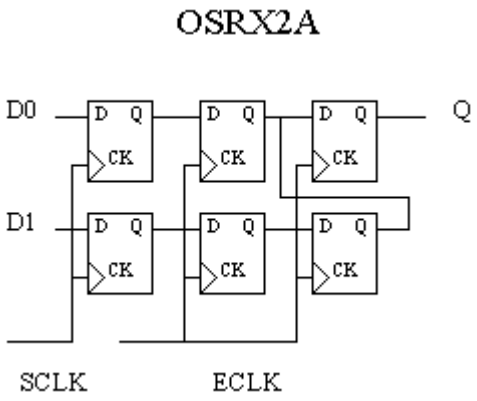


INPUTS: D0, D1, ECLK, SCLK, RST

OUTPUTS: Q, UPDATE

Description

Outputs data through the shift register to the output data. The following symbolic diagram shows the flip-flop structure of this primitive.

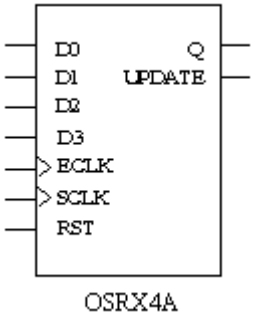


OSRX4A

Output 4-Bit Shift Register

Architectures Supported:

- LatticeSC/M

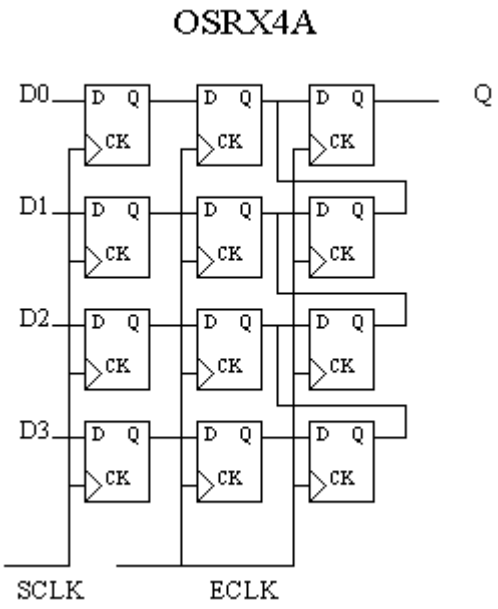


INPUTS: D0, D1, D2, D3, ECLK, SCLK, RST

OUTPUTS: Q, UPDATE

Description

Outputs data through the shift register to the output data. The following symbolic diagram shows the flip-flop structure of this primitive.



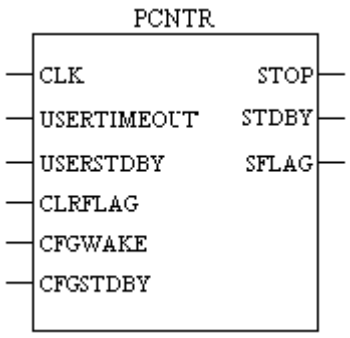
P

PCNTR

Power Controller

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLK, USERTIMEOUT, USERSTDBY, CLRFLAG, CFGWAKE, CFGSTDBY

OUTPUTS: STOP, STDBY, SFLAG

ATTRIBUTES:

STDBYOPT: "USER_CFG" (default), "USER", "CFG"

TIMEOUT: "BYPASS" (default), "USER", "COUNTER"

WAKEUP: "USER_CFG" (default), "USER", "CFG"

POROFF: "FALSE" (default), "TRUE"

BGOFF: "FALSE" (default), "TRUE"

Description

PCNTR is the MachXO2/Platform Manager 2 power controller primitive.

For more information, refer to the following technical note on the Lattice web site:

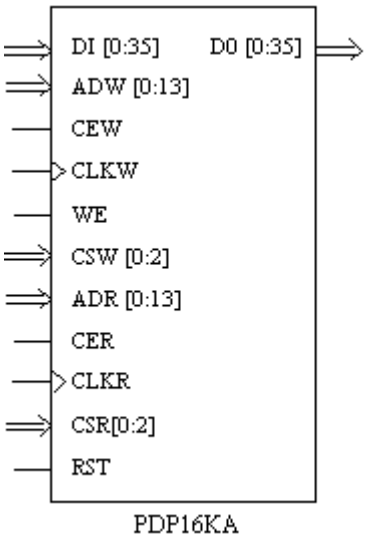
- TN1198 - Power Estimation and Management for MachXO2 Device

PDP16KA

16K Pseudo Dual Port Block RAM

Architectures Supported:

- LatticeSC/M



INPUTS: DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, DI18, DI19, DI20, DI21, DI22, DI23, DI24, DI25, DI26, DI27, DI28, DI29, DI30, DI31, DI32, DI33, DI34, DI35, ADW0, ADW1, ADW2, ADW3, ADW4, ADW5, ADW6, ADW7, ADW8, ADW9, ADW10, ADW11, ADW12, ADW13, CEW, CLKW, WE, CSW0, CSW1, CSW2, ADR0, ADR1, ADR2, ADR3, ADR4, ADR5, ADR6, ADR7, ADR8, ADR9, ADR10, ADR11, ADR12, ADR13, CER, CLKR, CSR0, CSR1, CSR2, RST

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17, DO18, DO19, DO20, DO21, DO22, DO23, DO24, DO25, DO26, DO27, DO28, DO29, DO30, DO31, DO32, DO33, DO34, DO35

ATTRIBUTES:

DATA_WIDTH_W: 1, 2, 4, 9, 18 (default), 36

DATA_WIDTH_R: 1, 2, 4, 9, 18 (default), 36

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 3-bit binary value (default: 3’b000)

CSDECODE_R: any 3-bit binary value (default: 3’b000)

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: (*Verilog*) 320'hXXX...X (80-bit hexadecimal value)
(*VHDL*) 0xXXX...X (80-bit hexadecimal value)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site on details of EBR port definition, attribute definition and usage.

- TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices

Note

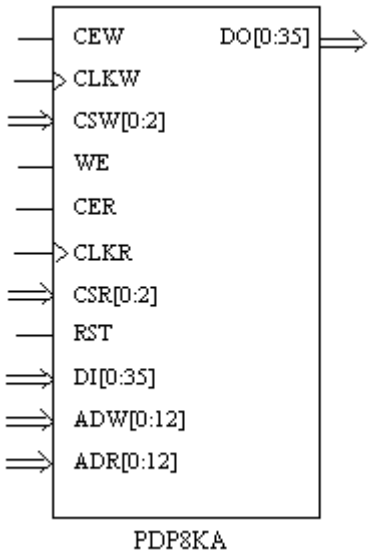
When the write data width (DATA_WIDTH_W) is set to 36, the WE port is invalid, that is, it has no effect on the data output.

PDP8KA

8K Pseudo Dual Port Block RAM

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: CEW, CLKW, CSW0, CSW1, CSW2, WE, CER, CLKR, CSR0, CSR1, CSR2, RST, DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, DI18, DI19, DI20, DI21, DI22, DI23, DI24, DI25, DI26, DI27, DI28, DI29, DI30, DI31, DI32, DI33, DI34, DI35, ADW0, ADW1, ADW2, ADW3, ADW4, ADW5, ADW6, ADW7, ADW8, ADW9, ADW10, ADW11, ADW12, ADR0, ADR1, ADR2, ADR3, ADR4, ADR5, ADR6, ADR7, ADR8, ADR9, ADR10, ADR11, ADR12

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17, DO18, DO19, DO20, DO21, DO22, DO23, DO24, DO25, DO26, DO27, DO28, DO29, DO30, DO31, DO32, DO33, DO34, DO35

ATTRIBUTES (LatticeXP/EC):

DATA_WIDTH_W: 1, 2, 4, 9, 18, 36 (default)

DATA_WIDTH_R: 1, 2, 4, 9, 18, 36 (default)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 3-bit binary value (default: 111)

CSDECODE_R: any 3-bit binary value (default: 111)

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_1F: (*Verilog*) 320'hXXX...X (80-bit hexadecimal value)
(*VHDL*) 0xXXX...X (80-bit hexadecimal value)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site on details of EBR port definition, attribute definition and usage.

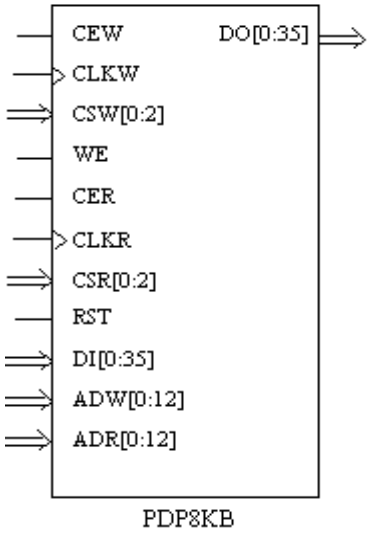
- TN1051 - Memory Usage Guide for LatticeECP/EC and LatticeXP Devices

PDP8KB

8K Pseudo Dual Port Block RAM

Architectures Supported:

- MachXO
- Platform Manager



INPUTS: CEW, CLKW, CSW0, CSW1, CSW2, WE, CER, CLKR, CSR0, CSR1, CSR2, RST, DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, DI18, DI19, DI20, DI21, DI22, DI23, DI24, DI25, DI26, DI27, DI28, DI29, DI30, DI31, DI32, DI33, DI34, DI35, ADW0, ADW1, ADW2, ADW3, ADW4, ADW5, ADW6, ADW7, ADW8, ADW9, ADW10, ADW11, ADW12, ADR0, ADR1, ADR2, ADR3, ADR4, ADR5, ADR6, ADR7, ADR8, ADR9, ADR10, ADR11, ADR12

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17, DO18, DO19, DO20, DO21, DO22, DO23, DO24, DO25, DO26, DO27, DO28, DO29, DO30, DO31, DO32, DO33, DO34, DO35

ATTRIBUTES:

DATA_WIDTH_W: 1, 2, 4, 9, 18 (default), 36

DATA_WIDTH_R: 1, 2, 4, 9, 18 (default), 36

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 3-bit binary value (default: 3’b000)

CSDECODE_R: any 3-bit binary value (default: 3’b000)

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_1F: (*Verilog*) "320'hXXX...X" (80-bit hex string)
(*VHDL*) "0xXXX...X" (80-bit hex string)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site on details of EBR port definition, attribute definition and usage.

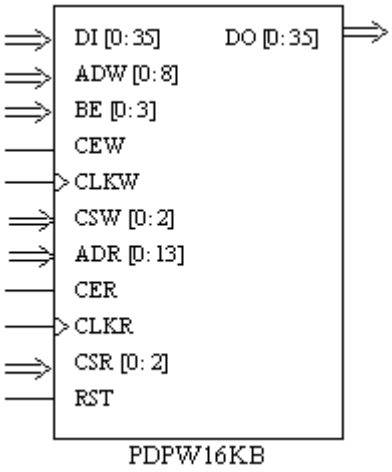
- TN1092 - MachXO Memory Usage Guide

PDPW16KB

Pseudo Dual Port Block RAM

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, DI18, DI19, DI20, DI21, DI22, DI23, DI24, DI25, DI26, DI27, DI28, DI29, DI30, DI31, DI32, DI33, DI34, DI35, ADW0, ADW1, ADW2, ADW3, ADW4, ADW5, ADW6, ADW7, ADW8, BE0, BE1, BE2, BE3, CEW, CLKW, CSW0, CSW1, CSW2, ADR0, ADR1, ADR2, ADR3, ADR4, ADR5, ADR6, ADR7, ADR8, ADR9, ADR10, ADR11, ADR12, ADR13, CER, CLKR, CSR0, CSR1, CSR2, RST

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17, DO18, DO19, DO20, DO21, DO22, DO23, DO24, DO25, DO26, DO27, DO28, DO29, DO30, DO31, DO32, DO33, DO34, DO35

ATTRIBUTES:

DATA_WIDTH_W: 1, 2, 4, 9, 18, 36 (default)

DATA_WIDTH_R: 1, 2, 4, 9, 18, 36 (default)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 3-bit binary value (default: 0b000)

CSDECODE_R: any 3-bit binary value (default: 0b000)

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: "0xXXX...X" (80-bit hex string) (default: all zeros)

Description

You can refer to the following technical notes on the Lattice web site on details of EBR port definition, attribute definition and usage.

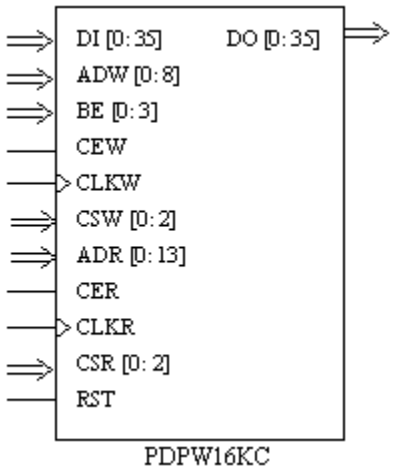
- TN1104 - LatticeECP2/M Memory Usage Guide
- TN1137 - LatticeXP2 Memory Usage Guide

PDPW16KC

Pseudo Dual Port Block RAM

Architectures Supported:

- LatticeECP3



INPUTS: DI35, DI34, DI33, DI32, DI31, DI30, DI29, DI28, DI27, DI26, DI25, DI24, DI23, DI22, DI21, DI20, DI19, DI18, DI17, DI16, DI15, DI14, DI13, DI12, DI11, DI10, DI9, DI8, DI7, DI6, DI5, DI4, DI3, DI2, DI1, DI0, ADW8, ADW7, ADW6, ADW5, ADW4, ADW3, ADW2, ADW1, ADW0, BE3, BE2, BE1, BE0, CEW, CLKW, CSW2, CSW1, CSW0, ADR13, ADR12, ADR11, ADR10,

ADR9, ADR8, ADR7, ADR6, ADR5, ADR4, ADR3, ADR2, ADR1, ADR0, CER, CLKR, CSR2, CSR1, CSR0, RST

OUTPUTS: DO35, DO34, DO33, DO32, DO31, DO30, DO29, DO28, DO27, DO26, DO25, DO24, DO23, DO22, DO21, DO20, DO19, DO18, DO17, DO16, DO15, DO14, DO13, DO12, DO11, DO10, DO9, DO8, DO7, DO6, DO5, DO4, DO3, DO2, DO1, DO0

ATTRIBUTES:

DATA_WIDTH_W: 1, 2, 4, 9, 18, 36 (default)

DATA_WIDTH_R: 1, 2, 4, 9, 18 (default), 36

REGMODE: "NOREG" (default), "OUTREG"

CSDECODE_W: any 3-bit binary value (default: all zeros)

CSDECODE_R: any 3-bit binary value (default: all zeros)

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: "0xXXX...X" (80-bit hex string) (default: all zeros)

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

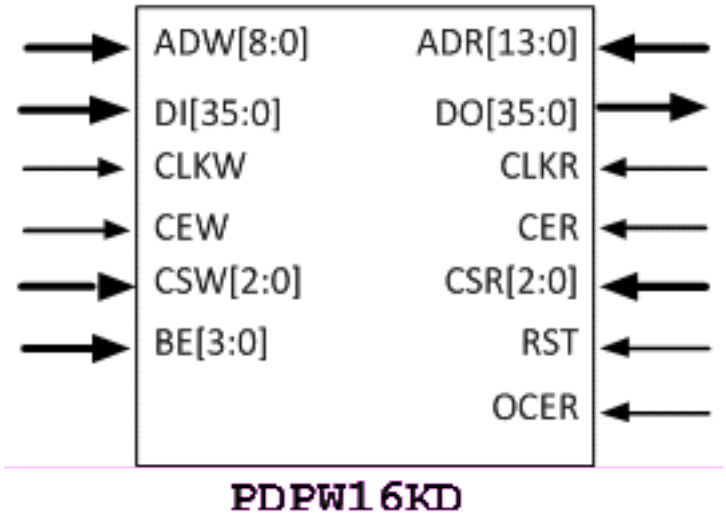
- TN1179 - LatticeECP3 Memory Usage Guide

PDPW16KD

Pseudo Dual Port RAM

Architectures Supported:

- ECP5



INPUTS: DI35, DI34, DI33, DI32, DI31, DI30, DI29, DI28, DI27, DI26, DI25, DI24, DI23, DI22, DI21, DI20, DI19, DI18, DI17, DI16, DI15, DI14, DI13, DI12, DI11, DI10, DI9, DI8, DI7, DI6, DI5, DI4, DI3, DI2, DI1, DI0, ADW8, ADW7, ADW6, ADW5, ADW4, ADW3, ADW2, ADW1, ADW0, ADR8, ADR7, ADR6, ADR5, ADR4, ADR3, ADR2, ADR1, ADR0, BE3, BE2, BE1, BE0, CEW, CLKW, CSW2, CSW1, CSW0, ADR13, ADR12, ADR11, ADR10, ADR9, ADR8, ADR7, ADR6, ADR5, ADR4, ADR3, ADR2, ADR1, ADR0, CER, CLKR, CSR2, CSR1, CSR0, OCER, RST

OUTPUTS: DO35, DO34, DO33, DO32, DO31, DO30, DO29, DO28, DO27, DO26, DO25, DO24, DO23, DO22, DO21, DO20, DO19, DO18, DO17, DO16, DO15, DO14, DO13, DO12, DO11, DO10, DO9, DO8, DO7, DO6, DO5, DO4, DO3, DO2, DO1, DO0

ATTRIBUTES:

The following table lists PDPW16KD Attributes:

Figure 62:

Description	Attribute Name	Attribute Type	Value	Default Value
Read Port Data Width	DATA_WIDTH_R	C	1, 2, 4, 9, 18, 36	36
Write Port Data Width	DATA_WIDTH_W	C	36	36
Enable Output Registers	REGMODE	C	NOREG, OUTREG	NOREG

Figure 62:

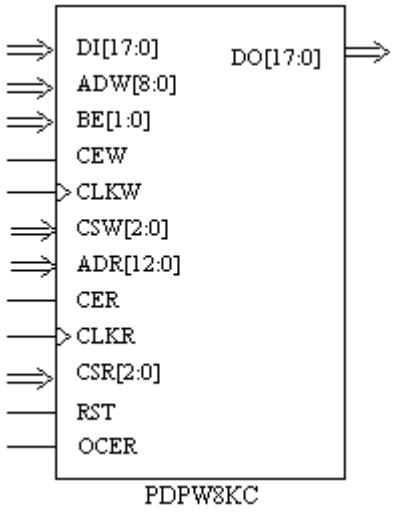
Description	Attribute Name	Attribute Type	Value	Default Value
Enable GSR	GSR	C	ENABLED, DISABLED	ENABLED
Reset Mode	RESETMODE	C	ASYNC, SYNC	SYNC
Reset Release	ASYNC_RESET_RELEASE	C	ASYNC, SYNC	SYNC
Memory File Format		C	BINARY, HEX, ADDRESSEDHEX	
Chip Select Decode for Port A	CSDECODE_W	C		0b000
Chip Select Decode for Port B	CSDECODE_R	C		0b000
Init Value	INITVAL_00… INITVAL_3F	C		
MEM_LPC_FILE	MEM_LPC_FILE	Core Only		
MEM_INIT_FILE	MEM_INIT_FILE	Core Only		
Defines if the memory file can be updated	INIT_DATA	C	STATIC, DYNAMIC	STATIC
Enable Output ClockEn	OCER	C	ENABLE, DISABLE	DISABLE
WID		Core Only		

PDPW8KC

Pseudo Dual Port Block RAM

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: DI17, DI16, DI15, DI14, DI13, DI12, DI11, DI10, DI9, DI8, DI7, DI6, DI5, DI4, DI3, DI2, DI1, DI0, ADW8, ADW7, ADW6, ADW5, ADW4, ADW3, ADW2, ADW1, ADW0, BE1, BE0, CEW, CLKW, CSW2, CSW1, CSW0, ADR12, ADR11, ADR10, ADR9, ADR8, ADR7, ADR6, ADR5, ADR4, ADR3, ADR2, ADR1, ADR0, CER, CLKR, CSR2, CSR1, CSR0, RST, OCER

OUTPUTS: DO17, DO16, DO15, DO14, DO13, DO12, DO11, DO10, DO9, DO8, DO7, DO6 DO5, DO4, DO3, DO2, DO1, DO0

ATTRIBUTES:

DATA_WIDTH_W: 18 (default)

DATA_WIDTH_R: 1, 2, 4, 9, 18 (default: 9)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE_W: any 3-bit binary value (default: all zeros)

CSDECODE_R: any 3-bit binary value (default: all zeros)

GSR: "ENABLED" (default), "DISABLED"

RESETMODE: "SYNC" (default), "ASYNC"

ASYNC_RESET_RELEASE: "SYNC" (default), "ASYNC"

INIT_DATA: "STATIC" (default), "DYNAMIC"

INITVAL_00 to INITVAL_1F: (*Verilog*) "320'hXXX...X" (80-bit hex string)
(*VHDL*) "0xXXX...X" (80-bit hex string)
Default: all zeros

Description

The following table describes the I/O ports of the PDPW8KC primitive.

Table 859:

Port Name	I/O	Definition
CLKR	I	Read clock
CLKW	I	Write clock
RST	I	Reset
CSW[1:0]	I	Chip select write
CSR[1:0]	I	Chip select read
CER	I	Read clock enable
CEW	I	Write clock enable
OCER	I	Read output clock enable
BE[1:0]	I	Byte enable
DI[17:0]	I	Write data (up to 18)
ADW[8:0]	I	Write address (up to 9)
ADR[12:0]	I	Read address (up to 13)
DO[17:0]	O	Read data (up to 18)

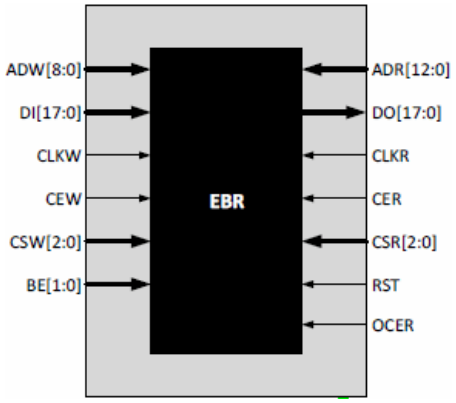
You can refer to the following technical note on the Lattice web site on details of EBR port definition, attribute definition and usage.

- TN1201 - Memory Usage Guide for MachXO2 Devices

PDPW8KE

Architectures Supported:

- LIFMD
- LIFMDF



INPUTS: DI17, DI16, DI15, DI14, DI13, DI12, DI11, DI10, DI9, DI8, DI7, DI6, DI5, DI4, DI3, DI2, DI1, DI0, BE1, BE0, ADW8, ADW7, ADW6, ADW5, ADW4, ADW3, ADW2, ADW1, ADW0, CEW, CLKW, CSW2, CSW1, CSW0, ADR12, ADR11, ADR10, ADR9, ADR8, ADR7, ADR6, ADR5, ADR4, ADR3, ADR2, ADR1, ADR0, CER, OCER, CLKR, CSR2, CSR1, CSR0, RST

OUTPUTS: DO17, DO16, DO15, DO14, DO13, DO12, DO11, DO10, DO9, DO8, DO7, DO6, DO5, DO4, DO3, DO2, DO1, DO0

.

Table 860:

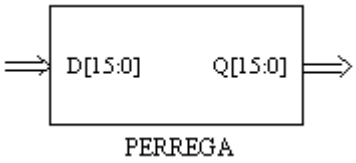
Port Name	Definition
CLKR	Read Clock
CLKW	Write Clock
RST	Reset
CSW[2:0]	Chip Select Write
CSR[2:0]	Chip Select Read
CER	Read Clock Enable
CEW	Write Clock Enable
OCER	Read Output Clock Enable
BE[1:0]	Byte Enable
{DIB[8:0], DIA[8:0]}	Write Data
ADW[8:0]	Write Address
ADW[12:0]	Read Address

PERREGA

Persistent User Register

Architectures Supported:

- LatticeECP3



INPUTS: D15, D14, D13, D12, D11, D10, D9, D8, D7, D6, D5, D4, D3, D2, D1, D0

OUTPUTS: Q15, Q14, Q13, Q12, Q11, Q10, Q9, Q8, Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0

Description

The PERREGA primitive enables you to use 16-bit registers to store information when the device goes into the configuration mode. The data on these registers will stay intact during the reconfiguration and be available for use after then. For example, you can use these registers to capture the last pattern that causes an error to reboot from the golden source when using SED.

These latches are available through the internal CIB. The below table describes the I/O ports of the PERREGA primitive.

Table 861:

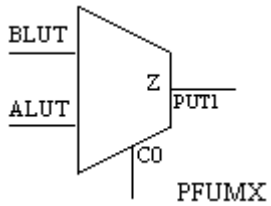
Signal	Type	Description	Function
D[15:0]	CIB Input	Parallel Data In	Provides parallel data from the FPGA fabric for the latch.
Q[15:0]	CIB Output	Parallel Data Out	Provides parallel data to the FPGA fabric for the latch.
PROGRAMN	Control Signal	High to Low Edge on the PROGRAMN pin	Latch in the data from D[15:0].
DONE	Internal Signal	Done signal	Prohibits multiple latching due to transitional glitches on the LE CIB or PROGRAMN pin.

PFUMX

2-Input Mux within the PFU, C0 Used for Selection with Positive Select

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: ALUT, BLUT, C0

OUTPUT: Z

Truth Table

Table 862:

INPUTS			OUTPUTS
BLUT	ALUT	C0	Z
X	1	1	1
X	0	1	0
1	X	0	1
0	X	0	0

X = Don’t care

Note

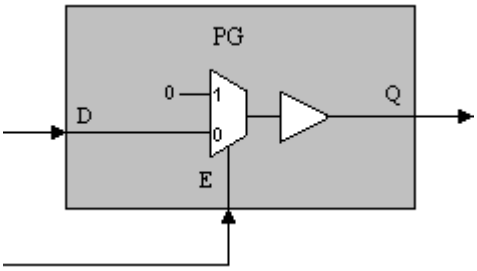
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

PG

Power Guard

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: D, E

OUTPUT: Q

Description

In the power guard mode, the receiver input is disconnected from the pad and held low with a weak pull down. The pad can be toggling and no receiver power will be used. Power guard is enabled on a bank by bank basis. Each bank has a CIB input signal to enable power guard. In addition, each PIO has individually programmable bit control to disable or enable the power guard capability.

PG is the power guard primitive for MachXO2/Platform Manager 2 devices. You can generate a primitive for a group of I/O pins and then connect it in the design. The PG primitive D port must be connected to the IO pin. You are required to connect the enable input of the primitive to the signal that is used to activate the power guard function. Only one enable signal can be used for all I/Os within an individual I/O bank using power guard. An individual I/O pin is only allowed to connect to one PG primitive module but there could be more

than one PG module active in a single I/O bank. Once an I/O signal is connected to the power guard primitive, the software sets the individually programmable bit for that PIO to enable power guard. The default setting for this bit is to disable the power guard function for a PIO.

The following table describes the IO ports of the PG primitive.

Table 863:

Port Name	I/O	Description
D	I	This is the signal coming from an input or I/O pad. This pin cannot have any fanout—only connects between pad and PG.
E	I	This is the “ENABLE” input that is tied to BIE (Block Input Enable). When E=0, Q is connected to D. When E=1, Q is isolated from D.
Q	O	This is the output of the power guard that drives towards the core. When E=0, the output Q is driven by the input D.

For more information, refer to the following technical note on the Lattice web site:

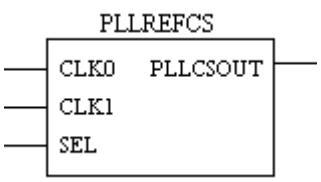
- TN1198 - Power Estimation and Management for MachXO2 Device

PLLREFCS

PLL Dynamic Reference Clock Switching

Architectures Supported:

- ECP5
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CLK0, CLK1, SEL

OUTPUT: PLLCSOUT

Description

The PLLREFCS primitive is used to support the dynamic reference clock switching using the SEL signal. This primitive does not require a wrapper.

You only need to instantiate the PLLREFCS primitive if you want to select between two different reference clocks in your application. If you only use one reference clock for the PLL, the software will automatically make the correct connections.

The following table describes the I/O ports of the PLLREFCS primitive.

Table 864:

Port Name	I/O	Description
CLK0, CLK1	I	There are two clock input MUXES (8 inputs each) which are labeled as REFCLK1 and REFCLK2. CLK0 is the output of REFCLK1. CLK1 is the output of REFCLK2.
SEL	I	Selects which signal goes to the input clock MUX.
PLLCSOUT	O	

For more information, refer to the following technical note on the Lattice web site:

- TN1199 - MachXO2 sysCLOCK PLL Design and Usage Guide

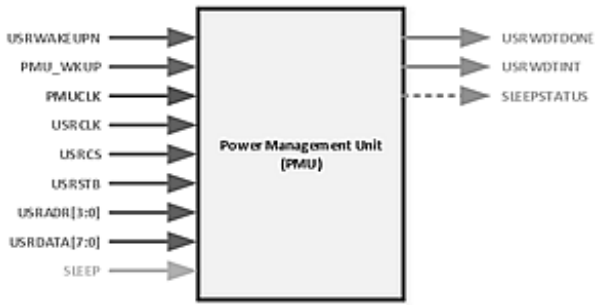
PMUA

Power Management Unit

The PMU user primitive serves as configurable HDL module that can be instantiated in user design to achieve power reduction for Lattice FPGA products. The primitive is treated as a black box in the software.

Architectures Supported:

- LIFMD
- LIFMDF



INPUTS: USRWAKEUP; I2CWAKEUP; INTCLK; EXTCLK; USRCLK; USRCS; USRSTB; USRADR3, USRADR2, USRADR1, USRADR0;

USRDATA7, USRDATA6, USRDATA5, USRDATA4, USRDATA3, USRDATA2, USRDATA1, USRDATA0; SLEEP

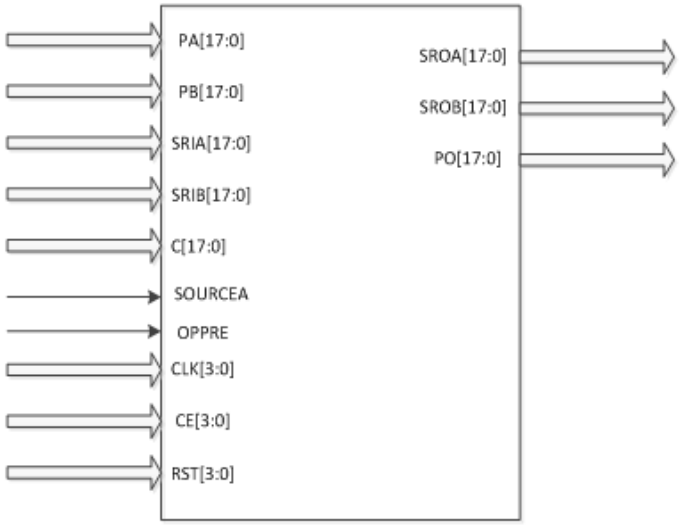
OUTPUT: USRWDTDONE; USRWDTINT; SLEEPSTATUS

PRADD18A

18-bit PreAdder/Shift

Architectures Supported:

- ECP5



PRADD18A

INPUTS: PA17, PA16, PA15, PA14, PA13, PA12, PA11, PA10, PA9, PA8, PA7, PA6, PA5, PA4, PA3, PA2, PA1, PA0, PB17, PB16, PB15, PB14, PB13, PB12, PB11, PB10, PB9, PB8, PB7, PB6, PB5, PB4, PB3, PB2, PB1, PB0, SRIA17, SRIA16, SRIA15, SRIA14, SRIA13, SRIA12, SRIA11, SRIA10, SRIA9,

SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB17, SRIB16, SRIB15, SRIB14, SRIB13, SRIB12, SRIB11, SRIB10, SRIB9, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0, C17, C16, C15, C14, C13, C12, C11, C10, C9, C8, C7, C6, C5, C4, C3, C2, C1, C0, SOURCEA, OPPRE, CLK3, CLK2, CLK1, CLK0, CE3, CE2, CE1, CE0, RST3, RST2, RST1, RST0

OUTPUT: SROA17, SROA16, SROA15, SROA14, SROA13, SROA12, SROA11, SROA10, SROA9, SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB17, SROB16, SROB15, SROB14, SROB13, SROB12, SROB11, SROB10, SROB9, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, PO17, PO16, PO15, PO14, PO13, PO12, PO11, PO10, PO9, PO8, PO7, PO6, PO5, PO4, PO3, PO2, PO1, PO0

ATTRIBUTES:

The attributes of PRADD18A are identical to the attributes of PRADD9A.

Description

The following table describes the I/O ports of the PRADD18A primitive.

Table 865:

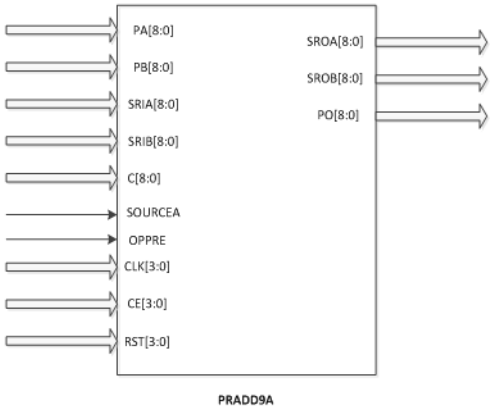
Port Name	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SOURCEA	I	Source Selector for Pre-adder Input A
PA[17:0]	I	Pre-adder Parallel Input A
PB[17:0]	I	Pre-adder Parallel Input B
SRIA[17:0]	I	Pre-adder Shift Input A
SRIB[17:0]	I	Pre-adder Shift Input A, backward direction
C[17:0]	I	Input used for high-speed option
SROA[17:0]	O	Pre-adder Shift Output A
SROB[17:0]	O	Pre-adder Shift Output B
PO[17:0]	O	Pre-adder Addition Output
OPPRE	I	Opcode for PreAdder

PRADD9A

9-bit PreAdder/Shift

Architectures Supported:

- ECP5



INPUTS: PA8, PA7, PA6, PA5, PA4, PA3, PA2, PA1, PA0, PB8, PB7, PB6, PB5, PB4, PB3, PB2, PB1, PB0, SRIA8, SRIA7, SRIA6, SRIA5, SRIA4, SRIA3, SRIA2, SRIA1, SRIA0, SRIB8, SRIB7, SRIB6, SRIB5, SRIB4, SRIB3, SRIB2, SRIB1, SRIB0, C8, C7, C6, C5, C4, C3, C2, C1, C0, SOURCEA, OPPRE, CLK3, CLK2, CLK1, CLK0, CE3, CE2, CE1, CE0, RST3, RST2, RST1, RST0

OUTPUT: SROA8, SROA7, SROA6, SROA5, SROA4, SROA3, SROA2, SROA1, SROA0, SROB8, SROB7, SROB6, SROB5, SROB4, SROB3, SROB2, SROB1, SROB0, PO8, PO7, PO6, PO5, PO4, PO3, PO2, PO1, PO0

ATTRIBUTES

The following table describes the attributes for PRADD9A primitive.

Table 866:

Attribute Name	Values	Default Value
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0

Table 866:

Attribute Name	Values	Default Value
REG_INPUTC_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE
REG_INPUTC_CE	CE0, CE1, CE2, CE3	CE0
REG_INPUTC_RST	RST0, RST1, RST2, RST3	RST0
REG_OPPRE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE
REG_OPPRE_CE	CE0, CE1, CE2, CE3	CE0
REG_OPPRE_RST	RST0, RST1, RST2, RST3	RST0
CLK0_DIV	ENABLED, DISABLED	ENABLED
CLK1_DIV	ENABLED, DISABLED	ENABLED
CLK2_DIV	ENABLED, DISABLED	ENABLED
CLK3_DIV	ENABLED, DISABLED	ENABLED
HIGHSPEED_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE
GSR	ENABLED, DISABLED	ENABLED
CAS_MATCH_REG	TRUE, FALSE	FALSE
SOURCEA_MODE	A_SHIFT, C_SHIFT, A_C_DYNAMIC, HIGHSPEED	A_SHIFT
SOURCEB_MODE	SHIFT, PARALLEL, INTERNAL	SHIFT
FB_MUX	SHIFT, SHIFT_BYPASS, DISABLED	SHIFT
RESETMODE	SYNC, ASYNC	SYNC
PRADD_LOC	0, 1	0
SYMMETRY_MODE	DIRECT, INTERNAL	DIRECT

PRADD_LOC = 0 indicates left location; 1 indicates the right location

*Symmetry_MODE: DIRECT = MUX_PA0 set to 1. INTERNAL = MUX_PA0 set to 0.

*For 1D FIR Filter operation, Symmetry_Mode will be set to INTERNAL. Left leg of PreAdder will get input from Reg 12 output and the right leg will get input from Reg 13 output.

*For 2D FIR Filter operation, Symmetry_Mode will be set to DIRECT. Left leg of PreAdder will get the direct input from MUIA0 and the right leg of the PreAdder will get MUIA0 delayed by 2 clock cycles (Reg 12 and Reg 13). In

this mode, external MUIB0 port will not be used. Also, SOURCEB_MODE must also be set to INTERNAL.

Details of SOURCEA_MODE Attribute

Table 867:

IPExpress Operation	SOURCEA_MODE Attribute	SOURCEA Port	Mc1_pa_mux3	Mc1_pa_mux4
Shift	A_SHIFT	1	00	01
A	A_SHIFT	0	00	00
C	C_SHIFT	0	01	00
A/C Dynamic	A_C_DYNAMIC	Live	10	00
HighspeedAC	HIGHSPEED	0	11	00
Dynamic Shift/A	A_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

Details of SOURCEB_MODE Attribute

Table 868:

SOURCEB_MODE Attribute	Operation (mc1_pa_b0 mux)
SHIFT	SRIB coming from the adjacent PREADDER on the right
PARALLEL	PB
INTERNAL	Output of Reg. 12

Details of FB_MUX Attribute

Table 869:

FB_MUX Attribute	Operation (MUX_FB0)
SHIFT	Output of Reg. 16
SHIFT_BYPASS	Output of Reg. 15
DISABLED	For placer only (PreAdder on the left side)

Description

The following table describes the I/O ports of the PRADD9A primitive.

Table 870:

Port Name	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SOURCEA	I	Source Selector for Pre-adder Input A
PA[8:0]	I	Pre-adder Parallel Input A
PB[8:0]	I	Pre-adder Parallel Input B
SRIA[8:0]	I	Pre-adder Shift Input A
SRIB[8:0]	I	Pre-adder Shift Input A, backward direction
C[8:0]	I	Input used for high-speed option
SROA[8:0]	O	Pre-adder Shift Output A
SROB[8:0]	O	Pre-adder Shift Output B
PO[8:0]	O	Pre-adder Addition Output
OPPRE	I	Opcode for PreAdder

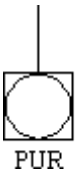
PUR

Power Up Set/Reset

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: PUR

ATTRIBUTES:

RST_PULSE: integer (default: 1)

Description

PUR is used to reset or set all register elements in your design upon device configuration/startup. The PUR component can be connected to a net from an input buffer or an internally generated net. It is active LOW and when pulsed will set or reset all register bits to the same state as the local set or reset functionality.

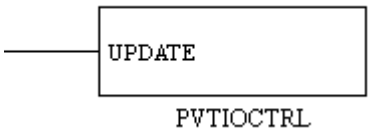
It is not necessary to connect signals for PUR to any register elements explicitly. The function will be implicitly connected globally.

PVTIOCTRL

PVT Monitor Circuit Controller

Architectures Supported:

- LatticeSC/M



INPUT: UPDATE

Description

The PVTIOCTRL primitive is used to generate a signal to control the PVT (Process, Voltage, and Temperature) monitor circuit. When UPDATE is 1, the PVT monitor circuit output will be updated. When UPDATE is 0, the last value of the PVT monitor circuit output is latched in.

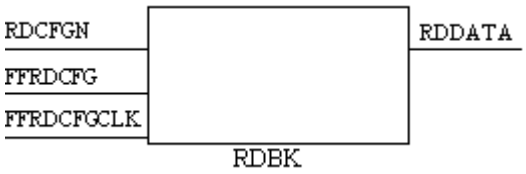
R

RDBK

Readback Controller

Architectures Supported:

- LatticeSC/M



INPUTS: RDCFGN, FFRDCFG, FFRDCFGCLK

OUTPUT: RDDATA

Description

RDBK is used to read back the configuration data and optionally the state of the PFU outputs. RDBK can be done while the FPGA is in normal system operation. To use RDBK, select options in the bit stream generator within the place and route tool.

- You can choose the option to prohibit readback, allow a single readback, or allow unrestricted readback. For more information on RDBK, refer to applicable technical notes or contact technical support.

RDCFGN: A high-to-low transition on this input initiates a readback operation. This pin must remain low during the readback operation.

RDDATA: Readback data is available at this output, which in turn is connected to the same pad as that used by TDO for boundary scan.

Note

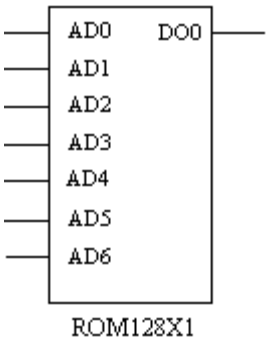
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

ROM128X1

128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: AD0, AD1, AD2, AD3, AD4, AD5, AD6

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 128'hXXX...X (32-bit hex string for LatticeECP2 and LatticeXP2; 32-bit hex value for other devices)
(*VHDL*) 0xXXX...X (32-bit hex string for LatticeECP2 and LatticeXP2; 32-bit hex value for other devices)
Default: all zeros

Description

The ROM128X1 symbol represents a 128 word by1-bit read-only memory. This ROM can be used to implement a ORCALUT7 in a design. The read operation is asynchronous and is always active. The memory is always being read.

The INITVAL=<*value*> attribute is used to initialize the ROM. The <*value*> should consist of 128 one-bit binary or 32 hexadecimal data written into the

ROM from the highest address to the lowest address. For example, if the following attribute is specified:

INITVAL=0x0123456789ABCDEF0123456789ABCDEF (in hex)

or

INITVAL=0000000100100011010001010110011111000100110101011111001101111011110000000100100011010001010110011111000100110101011110011101111011 (in binary)

it implies that the above data is loaded sequentially from location 127 to 0 (where location 127 would contain value 0 and location 0 value 1).

Truth Table

Table 871:

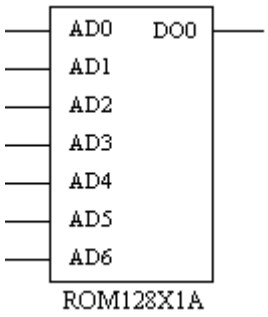
INPUTS	OUTPUTS	OPERATION
AD[5:0]	DO0	
AD[5:0]	MEM[AD[5:0]]	Read MEM[AD[5:0]]

ROM128X1A

128 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: AD6, AD5, AD4, AD3, AD2, AD1, AD0

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 128'hXXX...X (32-bit hexadecimal value)
(*VHDL*) 0xXXX...X (32-bit hexadecimal value)
Default: all zeros

Description

PFU based distributed 128 Word by 1 Bit ROM primitive. See Memory Primitives Overview for individual port description.

You can refer to the following technical notes on the Lattice web site for port definition, attribute definition and usage.

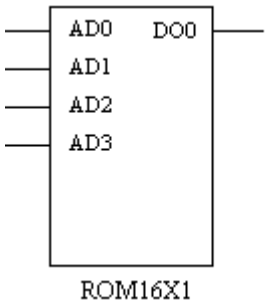
- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1179 - LatticeECP3 Memory Usage Guide

ROM16X1

16 Word by 1 Bit Read-Only Memory

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: AD0, AD1, AD2, AD3

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 16'hXXXX (4-bit hex string for LatticeECP2 and LatticeXP2; 4-bit hex value for other devices)
(*VHDL*) 0xXXXX (4-bit hex string for LatticeECP2 and LatticeXP2; 4-bit hex value for other devices)
Default: all zeros

Description

The ROM16X1 symbol represents a 16 word by 1 bit read-only memory. This ROM can be used to implement a ORCALUT4 in a design. The read operation is asynchronous and is always active. The memory is always being read.

The INITVAL=<*value*> attribute is used to initialize the ROM. The <*value*> should consist of 16 one-bit binary or 4 hexadecimal data written into the ROM from the highest address to the lowest address. For example, if the following attribute is specified:

INITVAL=0xF30A (in hex)

or

INITVAL=1111001100001010 (in binary)

it implies that the above data is loaded sequentially from location 15 to 0 (where location 15 would contain value 1 and location 0 value 0).

Truth Table

Table 872:

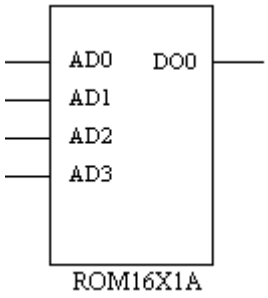
INPUTS	OUTPUTS	OPERATION
AD[3:0]	DO0	
AD[3:0]	MEM[AD[3:0]]	Read MEM[AD[3:0]]

ROM16X1A

16 Word by 1 Bit Read-Only Memory

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: AD3, AD2, AD1, AD0

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 16'hXXXX (4-bit hexadecimal value)
(*VHDL*) 0xXXXX (4-bit hexadecimal value)
Default: all zeros

Description

PFU based distributed 16 Word by 1 Bit ROM primitive. See Memory Primitives Overview for individual port description.

You can refer to the following technical notes on the Lattice web site for port definition, attribute definition, and usage.

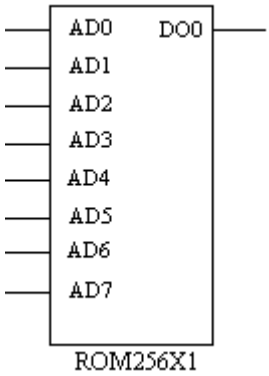
- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1179 - LatticeECP3 Memory Usage Guide

ROM256X1

256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: AD0, AD1, AD2, AD3, AD4, AD5, AD6, AD7

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 256'hXXX...X (64-bit hex string for LatticeECP2 and LatticeXP2; 64-bit hex value for other devices)
(*VHDL*) 0xXXX...X (64-bit hex string for LatticeECP2 and

LatticeXP2; 64-bit hex value for other devices)
Default: all zeros

Description

The ROM256X1 symbol represents a 256 word by1-bit read-only memory. This ROM can be used to implement a ORCALUT8 in a design. The read operation is asynchronous and is always active. The memory is always being read.

The INITVAL=<*value*> attribute is used to initialize the ROM. The <*value*> should consist of 256 one-bit binary or 64 hexadecimal data written into the ROM from the highest address to the lowest address. For example, if the following attribute is specified:

INITVAL=0x0123456789ABCDEF0123456789ABCDEF0123456789ABCDE
F0123456789ABCDEF (in hex)

or

INITVAL=0000000100100011010001010110011111000100110101011111001101111011100000001001000110100010101100111100010011010101111001101111011100000001001000110100010101100111100010011010101111001101111011100000001001000110100010101100111100010011010101111001101111011100000001001000110100010101100111100010011010101111001110111101111 (in binary)

it implies that the above data is loaded sequentially from location 255 to 0 (where location 255 would contain value 0 and location 0 value 1).

Truth Table

Table 873:

INPUTS	OUTPUTS	OPERATION
AD[5:0]	DO0	
AD[5:0]	MEM[AD[5:0]]	Read MEM[AD[5:0]]

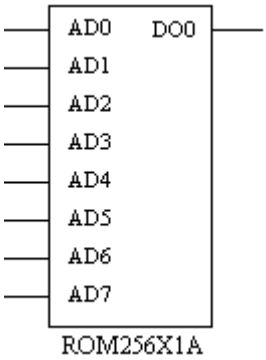
ROM256X1A

256 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF

- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: AD7, AD6, AD5, AD4, AD3, AD2, AD1, AD0

OUTPUTS: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 256'hXXX...X (64-bit hexadecimal value)
(*VHDL*) 0xXXX...X (64-bit hexadecimal value)
Default: all zeros

Description

PFU based distributed 256 Word by 1 Bit ROM primitive. See Memory Primitives Overview for individual port description.

You can refer to the following technical notes on the Lattice web site for port definition, attribute definition, and usage.

- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1179 - LatticeECP3 Memory Usage Guide

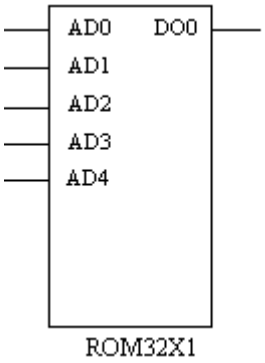
ROM32X1

32 Word by 1 Bit Read-Only Memory

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP

- LatticeXP2
- MachXO
- Platform Manager



INPUTS: AD0, AD1, AD2, AD3, AD4

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 32'hXXXXXXXX (8-bit hex string for LatticeECP2 and LatticeXP2; 8-bit hex value for other devices)
(*VHDL*) 0xXXXXXXXX (8-bit hex string for LatticeECP2 and LatticeXP2; 8-bit hex value for other devices)
Default: all zeros

Description

The ROM32X1 symbol represents a 32 word by 1 bit read-only memory. This ROM can be used to implement a ORCALUT5 in a design. The read operation is asynchronous and is always active. The memory is always being read.

The INITVAL=<*value*> attribute is used to initialize the ROM. The <*value*> should consist of 32 one-bit binary or 8 hexadecimal data written into the ROM from the highest address to the lowest address. For example, if the following attribute is specified:

INITVAL=0xF30A1234 (in hex)

or

INITVAL=11110011000010100001001000110100 (in binary)

it implies that the above data is loaded sequentially from location 31 to 0 (where location 31 would contain value 1 and location 0 value 0).

Truth Table

Table 874:

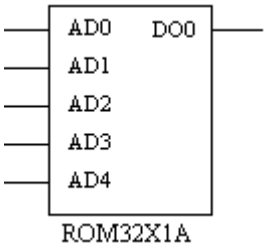
INPUTS	OUTPUTS	OPERATION
AD[4:0]	DO0	
AD[4:0]	MEM[AD[4:0]]	Read MEM[AD[4:0]]

ROM32X1A

32 Word by 1 Bit Read-Only Memory

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: AD4, AD3, AD2, AD1, AD0

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 32'hXXXXXXXX (8-bit hexadecimal value)
(*VHDL*) 0xXXXXXXXX (8-bit hexadecimal value)
Default: all zeros

Description

PFU based distributed 32 Word by 1 Bit ROM primitive. See Memory Primitives Overview for individual port description.

You can refer to the following technical notes on the Lattice web site for port definition, attribute definition, and usage.

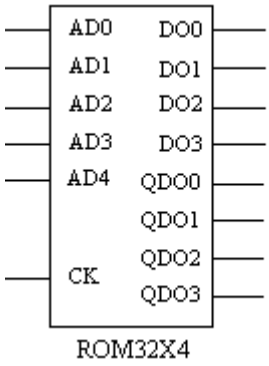
- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1179 - LatticeECP3 Memory Usage Guide

ROM32X4

32 Word by 4 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Architectures Supported:

- LatticeSC/M



INPUTS: AD0, AD1, AD2, AD3, AD4, CK

OUTPUTS: DO0, DO1, DO2, DO3, QDO0, QDO1, QDO2, QDO3

ATTRIBUTES:

INITVAL: (*Verilog*) 128'hXXX...X (32-bit hexadecimal value)
(*VHDL*) 0xXXX...X (32-bit hexadecimal value)
Default: all zeros

Description

The ROM32X4 symbol represents a 32 word by 4 bit read-only memory. The read operation is asynchronous and is always active. The memory is always being read.

This ROM also has registered data outputs (QDO[0:3]), which are registered on the rising edge of the clock.

The INITVAL=<*value*> attribute is used to initialize the ROM. The <*value*> should consist of 32 four-bit binary or 32 hexadecimal data written into the ROM from the highest address to the lowest address. For example, if the following attribute is specified:

INITVAL=0x0123456789ABCDEF0123456789ABCDEF (in hex)

or

INITVAL=00000001001000110100010101100111100010011010101111001101111011110000000100100011010001010110011111000100110101011110011101111101111 (in binary)

it implies that the above data is loaded sequentially from location 127 to 0 (where location 127 would contain value 0 and location 0 value 1).

Truth Table

Table 875:

INPUTS		OUTPUTS		OPERATION
AD[4:0]	CK	DO[3:0]	QDO[3:0]	
AD[4:0]	X	MEM[AD[4:0]]	QDO[3:0]	Read MEM[AD[4:0]]
AD[4:0]	↑	MEM[AD[4:0]]	MEM[AD[4:0]]	MEM[AD[4:0]] Register data outputs

X = Don’t care

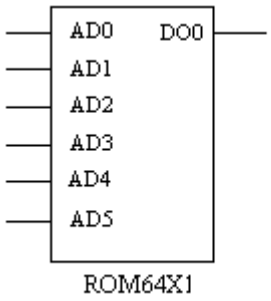
When GSR=0, QDO[3:0]=0

ROM64X1

64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M
- LatticeXP
- LatticeXP2
- MachXO
- Platform Manager



INPUTS: AD0, AD1, AD2, AD3, AD4, AD5

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 64'hXXXXXXXXXXXXXXXX (16-bit hex string for LatticeECP2 and LatticeXP2; 16-bit hex value for other devices)
(*VHDL*) 0xXXXXXXXXXXXXXXXX (16-bit hex string for LatticeECP2 and LatticeXP2; 16-bit hex value for other devices)
Default: all zeros

Description

The ROM64X1 symbol represents a 64 word by 1-bit read-only memory. This ROM can be used to implement a ORCALUT6 in a design. The read operation is asynchronous and is always active. The memory is always being read.

The INITVAL=<*value*> attribute is used to initialize the ROM. The <*value*> should consist of 64 one-bit binary or 16 hexadecimal data written into the ROM from the highest address to the lowest address. For example, if the following attribute is specified:

INITVAL=0x0123456789ABCDEF (in hex)

or

INITVAL=0000000100100011010001010110011110001001101010111100110111101111 (in binary)

it implies that the above data is loaded sequentially from location 63 to 0 (where location 63 would contain value 0 and location 0 value 1).

Truth Table

Table 876:

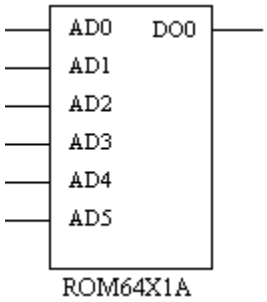
INPUTS	OUTPUTS	OPERATION
AD[5:0]	DO0	
AD[5:0]	MEM[AD[5:0]]	Read MEM[AD[5:0]]

ROM64X1A

64 Word by 1 Bit Positive Edge Triggered Read-Only Memory with Registered Data Outputs

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: AD5, AD4, AD3, AD2, AD1, AD0

OUTPUT: DO0

ATTRIBUTES:

INITVAL: (*Verilog*) 64'hXXXXXXXXXXXXXXXX (16-bit hexadecimal value)
(*VHDL*) 0xXXXXXXXXXXXXXXXX (16-bit hexadecimal value)
Default: all zeros

Description

PFU based distributed 64 Word by 1 Bit ROM primitive. See Memory Primitives Overview for individual port description.

You can also refer to the following technical notes on the Lattice web site for port definition, attribute definition, and usage.

- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1179 - LatticeECP3 Memory Usage Guide

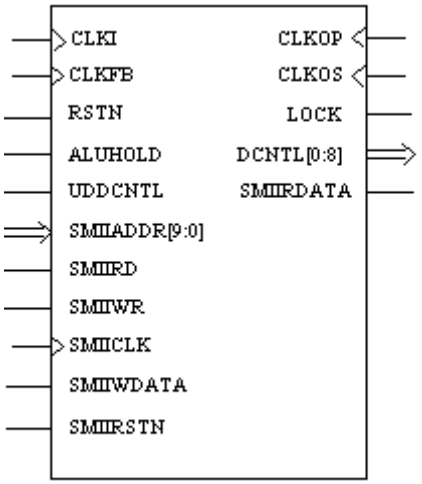
S

SDCDLLA

Single Delay Cell DLL

Architectures Supported:

- LatticeSC/M



SDCDLLA

INPUTS: CLKI, CLKFB, RSTN, ALUHOLD, UDDCNTL, SMIADDR9, SMIADDR8, SMIADDR7, SMIADDR6, SMIADDR5, SMIADDR4, SMIADDR3, SMIADDR2, SMIADDR1, SMIADDR0, SMIRD, SMIWR, SMICLK, SMIWDATA, SMIRSTN

OUTPUTS: CLKOP, CLKOS, LOCK, DCNTL0, DCNTL1, DCNTL2, DCNTL3, DCNTL4, DCNTL5, DCNTL6, DCNTL7, DCNTL8, SMIRDATA

ATTRIBUTES:

CLKOS_FPHASE: 0 (default), 11, 22, 45

CLKI_DIV: 1 (default), 2, 4

CLKOS_DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

CLKOS_FDEL_ADJ: "DISABLED" (default), "ENABLED'

ALU_LOCK_CNT: integers 3~15 (default: 3)

ALU_UNLOCK_CNT: integers 3~15 (default: 3)

GLITCH_TOLERANCE: integers 0~7 (default: 0)

DCNTL_ADJVAL: integers -127~127 (default: 0)

ALU_INIT_CNTVAL: 0, 4, 8, 12, 16, 32, 48, 64, 72 (default: 0)

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

SMI_OFFSET: 0x400~0x7FF (default: 12'h410)

MODULE_TYPE: "SDCDLLA"

IP_TYPE: "SDCDLLA"

Description

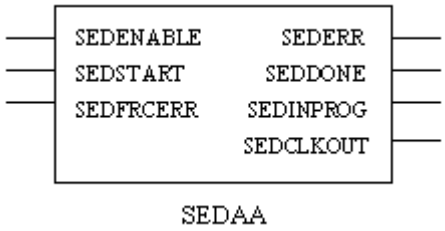
The single delay cell corrects for clock injection and enables the 9-bit ALU output. The primitive features a single clock output, lock achieved starting from minimum delay, output control bits, and allows +/- delay on output control bits. Its requirements are a maximum frequency of 700 MHz and a minimum frequency of 300 MHz.

SEDAA

SED (Soft Error Detect) Basic

Architectures Supported:

- LatticeECP2/M



INPUTS: SEDENABLE, SEDSTART, SEDFRCERR

OUTPUTS: SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT

ATTRIBUTES:

OSC_DIV: 1 (default), 2, 4, 8, 16, 32, 64, 128, 256

CHECKALWAYS: "DISABLED" (default), "ENABLED"

MCCLK_FREQ: "2.5" (default), "4.3", "5.4", "6.9", "8.1", "9.2", "10.0", "13", "15", "20", "26", "30", "34", "41", "45", "55", "60", "130"

DEV_DENSITY: (ECP2) "35K" (default), "6K", "12K", "20K", "50K", "70K"

DEV_DENSITY: (ECP2M) "M35K" (default), "M20K", "M50K", "M70K", "M100K"

ENCRYPTION: "OFF" (default), "ON"

Description

SEDAA is the basic SED primitive for LatticeECP2 devices. Soft errors occur when high-energy charged particles alter the stored charge in a memory cell in an electronic circuit. There are customer selectable software features in Lattice Diamond to support Soft Error Detect (SED) IP features. This is only applicable to devices that support SED in their architecture.

This primitive should be instantiated in the user's source code in VHDL or Verilog.

See the following table for port definition.

Table 877:

Port Name	I/O	Description
SEDSTART	I	Error detection start (sampled at positive clock edge).
SEDENABLE	I	SED enable (a Low clears the SED).
SEDFRCERR	I	Force an SED error flag (e.g. for testing), active high.
SEDINPROG	O	SED cycle is in progress, asserts high.
SEDDONE	O	SED cycle is complete, asserts high.
SEDERR	O	SED error flag, asserts high.
SEDCLKOUT	O	Optional clock output.

You can refer to the following technical note on the Lattice web site for more information and usage.

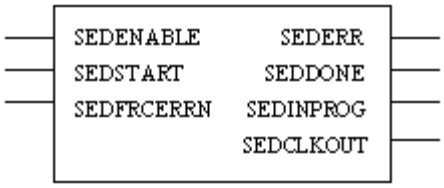
- TN1113 - LatticeECP2/M Soft Error Detection (SED) Usage Guide

SEDBA

Basic SED (Soft Error Detect)

Architectures Supported:

- LatticeXP2



SEDBA

INPUTS: SEDENABLE, SEDSTART, SEDFRCERRN

OUTPUTS: SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT

ATTRIBUTES:

OSC_DIV: 1 (default), 2, 4, 8, 16, 32, 64, 128, 256

CHECKALWAYS: "DISABLED" (default), "ENABLED"

MCCLK_FREQ: "3.1" (default), "2.5", "4.3", "5.4", "6.9", "8.1", "9.2", "10.0", "13", "15", "26", "32", "40", "54"

DEV_DENSITY: "17K" (default), "5K", "8K", "30K", "40K"

BOOT_OPTION: "INTERNAL" (default), "EXTERNAL"

Description

SEDBA is the basic soft error detect (SED) primitive for LatticeXP2 devices. Soft errors occur when high-energy charged particles alter the stored charge in a memory cell in an electronic circuit. There are customer selectable software features in Lattice Diamond to support SED IP features. This is only applicable to devices that support SED in their architecture. LatticeXP2devices have built-in SED circuitry to detect soft errors.

The SED primitive should be instantiated in user's VHDL or Verilog source code. There are two types of LatticeXP2 modules: Basic SED (SEDBA) and One Shot SED (SEDBB).

See the following table for port description

Table 878:

Port Name	I/O	Description
SEDSTART	I	Error detection start (sampled at positive clock edge).
SEDENABLE	I	SED enable (a Low clears the SED).
SEDFRCERRN	I	Force an SED error flag (e.g. for testing), active low.
SEDINPROG	O	SED cycle is in progress, asserts high.
SEDDONE	O	SED cycle is complete, asserts high.

Table 878:

Port Name	I/O	Description
SEDERR	O	SED error flag, asserts high.
SEDCLKOUT	O	Optional clock output.

You can refer to the following technical note on the Lattice web site for more information and usage.

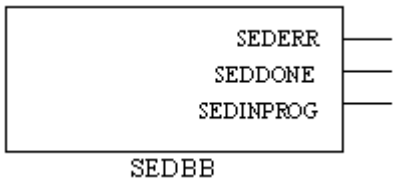
- TN1130 - LatticeXP2 Soft Error Detection (SED) Usage Guide

SEDBB

One Shot SED (Soft Error Detect)

Architectures Supported:

- LatticeXP2



OUTPUTS: SEDERR, SEDDONE, SEDINPROG

Description

SEDBB is the one shot soft error detect (SED) primitive for LatticeXP2 devices. Soft errors occur when high-energy charged particles alter the stored charge in a memory cell in an electronic circuit. There are customer selectable software features in Lattice Diamond to support SED IP features. This is only applicable to devices that support SED in their architecture. LatticeXP2 devices have built-in SED circuitry to detect soft errors.

The SED primitive should be instantiated in user's VHDL or Verilog source code. There are two types of LatticeXP2 modules: Basic SED (SEDBA) and One Shot SED (SEDBB).

See the following table for port description.

Table 879:

Port Name	I/O	Description
SEDINPROG	O	SED cycle is in progress, asserts high.
SEDDONE	O	SED cycle is complete, asserts high.
SEDERR	O	SED error flag, asserts high.

You can refer to the following technical note on the Lattice web site for more information and usage.

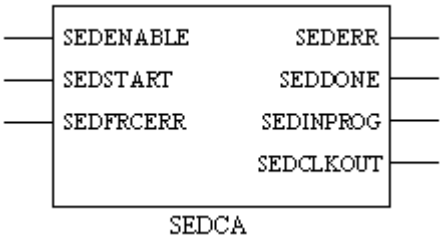
- TN1130 - LatticeXP2 Soft Error Detection (SED) Usage Guide

SEDCA

Basic SED (Soft Error Detect)

Architectures Supported:

- LatticeECP3



INPUTS: SEDENABLE, SEDSTART, SEDFRCERR

OUTPUTS: SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT

ATTRIBUTES:

OSC_DIV: 1 (default), 2, 4, 8, 16, 32, 64, 128, 256

CHECKALWAYS: "DISABLED" (default), "ENABLED"

MCCLK_FREQ: "2.5" (default), "4.3", "5.4", "6.9", "8.1", "9.2", "10.0", "13", "15", "20", "26", "30", "34", "41", "45", "55", "60", "130"

DEV_DENSITY: "95K" (default)

Description

SEDCA is the basic Soft Error Detect primitive. Basic SED runs on all bits only. It checks the CRC for all the bitstreams that include both “Care” and “Don’t Care” bits.

See the following table for port definition.

Table 880:

Port Name	I/O	Description
SEDENABLE	I	SED enable (a Low clears the SED).
SEDSTART	I	Error detection start (sampled at positive clock edge).

Table 880:

Port Name	I/O	Description
SEDFRCERR	I	Force an SED error flag (e.g. for testing), active high.
SEDINPROG	O	SED cycle is in progress, asserts High.
SEDDONE	O	SED cycle is complete, asserts High.
SEDERR	O	SED error flag, asserts High.
SEDCLKOUT	O	Optional clock output.

You can refer to the following technical note on the Lattice web site for more information and usage.

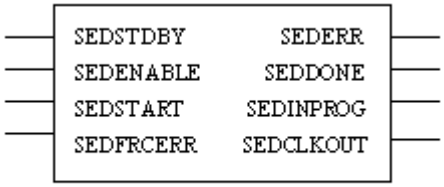
- TN1184 - LatticeECP3 Soft Error Detection (SED) Usage Guide

SEDFA

Soft Error Detect in Basic Mode

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



SEDFA

INPUTS: SEDSTDBY, SEDENABLE, SEDSTART, SEDFRCERR

OUTPUTS: SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT

ATTRIBUTES:

SED_CLK_FREQ: "2.08", "2.15", "2.22", "2.29", "2.38", "2.46", "2.56", "2.66", "2.77", "2.89", "3.02", "3.17", "3.33", "3.5" (default), "3.69", "3.91", "4.16", "4.29", "4.43", "4.59", "4.75", "4.93", "5.12", "5.32", "5.54", "5.78", "6.05", "6.33", "6.65", "7", "7.39", "7.82", "8.31", "8.58", "8.87", "9.17", "9.5", "9.85", "10.23", "10.64", "11.08", "11.57", "12.09", "12.67", "13.3", "14", "14.78",

"15.65", "16.63", "17.73", "19", "20.46", "22.17", "24.18", "26.6", "29.56", "33.25", "38", "44.33", "53.2", "66.5", "88.67", "133"

CHECKALWAYS: "DISABLED" (default), "ENABLED"

DEV_DENSITY:

MachXO2: "256L", "640L", "1200L", "2000L", "4000L", "7000L", "10000L", "640U", "1200U", "2000U", "4000U"

LFMNX, MachXO3D: "4300L", "9400L"

MachXO3L: "640L_121P", "1300L", "2100L", "4300L", "1300L_256P", "2100L_324P", "4300L_400P", "6900L"

Description

The Soft-Error Detect (SED) circuitry provides a method for the device to check its configuration data for soft-errors. SEDFA is used for SED basic mode.

See the following table for port description.

Table 881:

Port Name	I/O	Description
SEDSTDBY	I	SED standby.
SEDENABLE	I	SED enable (a Low clears the SED).
SEDSTART	I	Error detection start (sampled at positive clock edge).
SEDFRCERR	I	Force an SED error flag (e.g. for testing), active high.
SEDINPROG	O	SED cycle is in progress, asserts High.
SEDDONE	O	SED cycle is complete, asserts High.
SEDERR	O	SED error flag, asserts High.
SEDCLKOUT	O	Optional clock output.

You can refer to the following technical note on the Lattice web site for more information and usage.

- TN1206 - MachXO2 Soft Error Detection (SED) Usage Guide

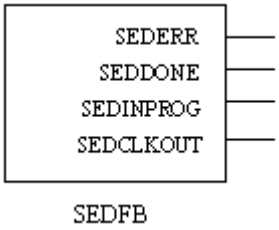
SEDFB

Soft Error Detect in One Shot Mode

Architectures Supported:

- LFMNX
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager 2



OUTPUTS: SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT

Description

The Soft-Error Detect (SED) circuitry provides a method for the device to check its configuration data for soft-errors. SEDFB is used for One Shot SED mode.

See the following table for port description.

Table 882:

Port Name	I/O	Description
SEDINPROG	O	SED cycle is in progress, asserts High.
SEDDONE	O	SED cycle is complete, asserts High.
SEDERR	O	SED error flag, asserts High.
SEDCLKOUT	O	Optional clock output.

You can refer to the following technical note on the Lattice web site for more information and usage.

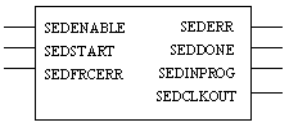
- TN1206 - MachXO2 Soft Error Detection (SED) Usage Guide

SEDGA

Soft Error Detect

Architectures Supported:

- ECP5



INPUTS: SEDENABLE, SEDSTART, SEDFRCERR

OUTPUTS: SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT

Description

The Soft-Error Detect (SED) circuitry provides a method for the device to check its configuration data for soft-errors.

See the following table for port description.

Table 883:

Port Name	I/O	Description
SEDENABLE	I	SED enable (a Low clears the SED).
SEDSTART	I	Error detection start (sampled at positive clock edge).
SEDFRCERR	I	Force an SED error flag (e.g. for testing), active high.
SEDINPROG	O	SED cycle is in progress, asserts High.
SEDDONE	O	SED cycle is complete, asserts High.
SEDERR	O	SED error flag, asserts High.
SEDCLKOUT	O	Optional clock output.

You can refer to the following technical note on the Lattice web site for more information and usage.

- TN1268 - ECP5 Soft Error Detection (SED) Usage Guide

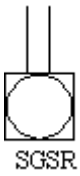
SGSR

Synchronous Release Global Set/Reset Interface

Architectures Supported:

- ECP5
- LFMNX
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3

- LatticeSC/M
- LatticeXP
- LatticeXP2
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: GSR, CLK

Description

SGSR is used to reset or set all register elements in your design. The SGSR component can be connected to a net from an input buffer or an internally generated net. It is active LOW and when pulsed will set or reset all flip-flops, latches, registers, and counters to the same state as the local set or reset functionality. When input GSR is HIGH, the global signal is released at the positive edge of the clock (CLK).

It is not necessary to connect signals for SGSR to any register elements explicitly. The function will be implicitly connected globally. The functionality of the SGSR for sequential cells without a local set or reset are described in the appropriate library help page.

Note

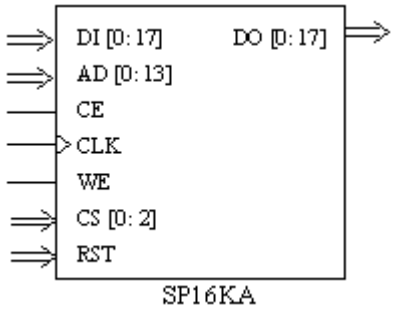
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

SP16KA

16K Single Port Block RAM

Architectures Supported:

- LatticeSC/M



INPUTS: DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, AD0, AD1, AD2, AD3, AD4, AD5, AD6, AD7, AD8, AD9, AD10, AD11, AD12, AD13, CE, CLK, WE, CS0, CS1, CS2, RST

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17

ATTRIBUTES:

DATA_WIDTH: 1, 2, 4, 9, 18 (default)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE: any 3-bit binary value (default: 3’b000)

WRITEMODE: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: (*Verilog*) 320'hXXX...X (80-bit hexadecimal value)
(*VHDL*) 0xXXX...X (80-bit hexadecimal value)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

- TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices

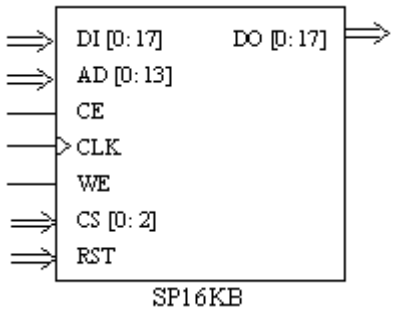
SP16KB

Single Port Block RAM

Architectures Supported:

- LatticeECP2/M

- LatticeXP2



INPUTS: DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, AD0, AD1, AD2, AD3, AD4, AD5, AD6, AD7, AD8, AD9, AD10, AD11, AD12, AD13, CE, CLK, WE, CS0, CS1, CS2, RST

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17

ATTRIBUTES:

DATA_WIDTH:1, 2, 4, 9, 18 (default)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE: any 3-bit binary value (default: 0b000)

WRITEMODE: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: "0xXXX...X" (80-bit hex string) (default: all zeros)

Description

Single Port Block RAM primitive. See Memory Primitives Overview for more information.

You can also refer to the following technical notes on the Lattice web site for port definition, attributes and usage.

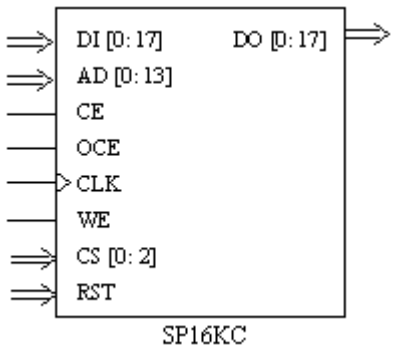
- TN1104 - LatticeECP2/M Memory Usage Guide
- TN1137 - LatticeXP2 Memory Usage Guide

SP16KC

Single Port Block RAM

Architectures Supported:

- LatticeECP3



INPUTS: DI17, DI16, DI15, DI14, DI13, DI12, DI11, DI10, DI9, DI8, DI7, DI6, DI5, DI4, DI3, DI2, DI1, DI0, AD13, AD12, AD11, AD10, AD9, AD8, AD7, AD6, AD5, AD4, AD3, AD2, AD1, AD0, CE, OCE, CLK, WE, CS2, CS1, CS0, RST

OUTPUTS: DO17, DO16, DO15, DO14, DO13, DO12, DO11, DO10, DO9, DO8, DO7, DO6, DO5, DO4, DO3, DO2, DO1, DO0

ATTRIBUTES:

DATA_WIDTH: 1, 2, 4, 9, 18 (default)

REGMODE: "NOREG" (default), "OUTREG"

CSDECODE: any 3-bit binary string (default: "0b000")

WRITEMODE: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_3F: "0xXXX...X" (80-bit hex string) (default: all zeros)

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

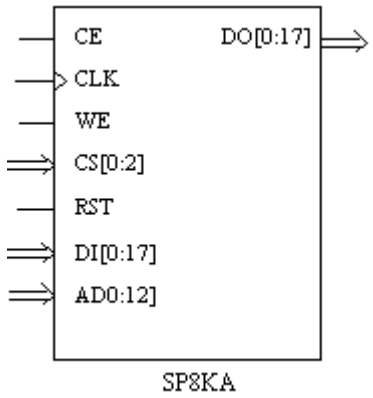
- TN1179 - LatticeECP3 Memory Usage Guide

SP8KA

8K Single Port Block RAM

Architectures Supported:

- LatticeECP/EC
- LatticeXP



INPUTS: CE, CLK, WE, CS0, CS1, CS2, RST, DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, AD0, AD1, AD2, AD3, AD4, AD5, AD6, AD7, AD8, AD9, AD10, AD11, AD12

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17

ATTRIBUTES:

REGMODE: "NOREG" (default), "OUTREG"

GSR: "DISABLED" (default), "ENABLED"

WRITEMODE: "NORMAL" (default), "WRITETHROUGH"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE: any 3-bit binary value (default: 111)

DATA_WIDTH: 1, 2, 4, 9, 18 (default)

INITVAL_00 to INITVAL_1F: (*Verilog*) 320'hXXX...X (80-bit hexadecimal value)
(*VHDL*) 0xXXX...X (80-bit hexadecimal value)
Default: all zeros

Description

You can refer to the following technical note on the Lattice web site for EBR port definition, attribute definition and usage.

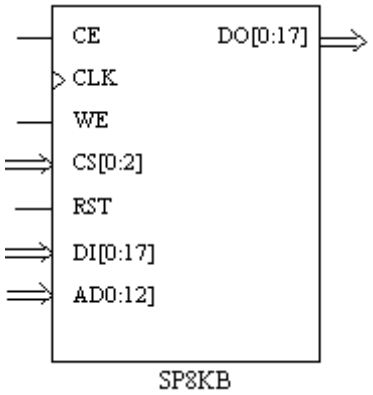
- TN1051 - Memory Usage Guide for LatticeECP/EC and LatticeXP Devices

SP8KB

8K Single Port Block RAM

Architectures Supported:

- MachXO
- Platform Manager



INPUTS: CE, CLK, WE, CS0, CS1, CS2, RST, DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, AD0, AD1, AD2, AD3, AD4, AD5, AD6, AD7, AD8, AD9, AD10, AD11, AD12

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17

ATTRIBUTES:

DATA_WIDTH: 1, 2, 4, 9, 18 (default)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE: any 3-bit binary value (default: 3’b000)

WRITEMODE: "NORMAL" (default), "WRITETHROUGH"

GSR: "DISABLED" (default), "ENABLED"

INITVAL_00 to INITVAL_1F: (*Verilog*) "320'hXXX...X" (80-bit hex string)
(*VHDL*) "0xXXX...X" (80-bit hex string)
Default: all zeros

Description

8K Single Port Block RAM primitive. See Memory Primitives Overview for more information.

You can also refer to the following technical note on the Lattice web site for port definition, attributes and usage.

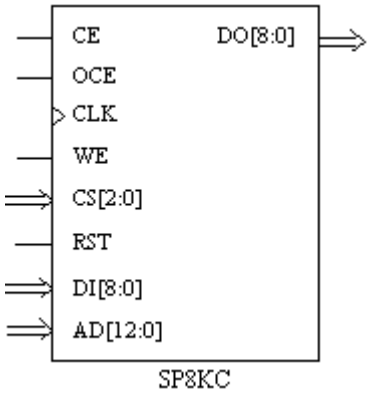
- TN1092 - MachXO Memory Usage Guide

SP8KC

8K Single Port Block RAM

Architectures Supported:

- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: CE, OCE, CLK, WE, CS0, CS1, CS2, RST, DI0, DI1, DI2, DI3, DI4, DI5, DI6, DI7, DI8, DI9, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DI17, AD0, AD1, AD2, AD3, AD4, AD5, AD6, AD7, AD8, AD9, AD10, AD11, AD12

OUTPUTS: DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7, DO8, DO9, DO10, DO11, DO12, DO13, DO14, DO15, DO16, DO17

ATTRIBUTES:

DATA_WIDTH: 1, 2, 4, 9 (default)

REGMODE: "NOREG" (default), "OUTREG"

RESETMODE: "SYNC" (default), "ASYNC"

CSDECODE: any 3-bit binary value (default: all zeros)

WRITEMODE: "NORMAL" (default), "WRITETHROUGH", "READBEFOREWRITE"

GSR: "ENABLED" (default), "DISABLED"

INIT_DATA: "STATIC" (default), "DYNAMIC"

INITVAL_00 to INITVAL_1F: (*Verilog*) "320'hXXX...X" (80-bit hex string)
(*VHDL*) "0xXXX...X" (80-bit hex string)
Default: all zeros

Description

8K Single Port Block RAM primitive. See the below table for I/O port description.

Table 884:

Port Name	I/O	Definition
CLK	I	Clock
CE	I	Clock enable
OCE	I	Output clock enable
AD[12:0]	I	Address bus
DI[8:0]	I	Input data
WE	I	Write enable
CS[2:0]	I	Chip select
RST	I	Reset
DO[8:0]	O	Output data

You can also refer to the following technical note on the Lattice web site for port definition, attributes and usage.

- TN1201 - Memory Usage Guide for MachXO2 Devices

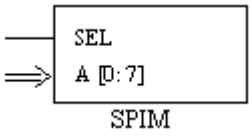
SPIM

SPIM Primitive Distributed RAM

Architectures Supported:

- LatticeECP2/M

- LatticeECP3



INPUTS: SEL, A0, A1, A2, A3, A4, A5, A6, A7

Description

Some devices have a built-in SPI CIB interface that is embedded into the hardware. The SPI CIB interface allows the user to control the dual-boot flash support based on the user's logic. Users can interface with this hardwired SPI controller through the use of the SPIM primitive. The SPIM primitive have 9 input ports only. The SEL line indicates which SPI Flash device to boot from and the signals [A0..A7] indicate the address from where the device will reboot during reconfiguration.

For further information on SPIm Mode, refer to the following technical notes on the Lattice web site:

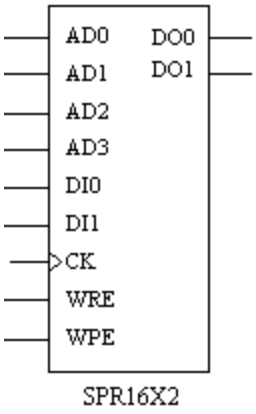
- TN1169 - LatticeECP3 sysCONFIG Usage Guide
- TN1108 - LatticeECP2/M sysCONFIG Usage Guide

SPR16X2

Distributed Single Port RAM

Architectures Supported:

- LatticeSC/M



INPUTS: AD0, AD1, AD2, AD3, DI0, DI1, CK, WRE, WPE

OUTPUTS: DO0, DO1

ATTRIBUTES:

INITVAL: (*Verilog*) 64'hXXXXXXXXXXXXXXXX (16-bit hexadecimal value)
(*VHDL*) 0xXXXXXXXXXXXXXXXX (16-bit hexadecimal value)
Default: all zeros

GSR: "ENABLED" (default), "DISABLED"

Description

The SPR16X2 symbol represents a 16 word by 2 bit asynchronous single port RAM. It has two data inputs DI[1:0], a positive Write Enable (WRE), one positive Write Port Enables (WPE), and one set of address inputs.

The WRE and WPE must be HIGH for the rising clock edge if the write to the RAM is occurring on the falling edge. The data is written into the locations specified by the write address lines AD[3:0] on the next negative clock (CK) edge. The data read operation is always performed asynchronously, with the memory contents specified by the address inputs AD[3:0] output on the data output signals DO[1:0].

In other words, the read operation is asynchronous and is always active. The write operation is synchronous and only occurs when there is a falling edge of the clock and the write enables are high prior to that falling edge.

If desired, the contents of the SPR16X2 can be assigned an initial value, which is loaded into the RAM during configuration. The INITVAL=<*value*> attribute is used to assign the initial value. The <*value*> should consist of 16 hexadecimal data written into the RAM from the highest address to the lowest address. For example, if the following attribute is specified:

INITVAL=0x0123012301230123 (in hex)

it implies that the above data is loaded sequentially from location FH to 0H (where FH would contain the 0 and 0H the 3). If no INITVAL attribute is specified, the RAM is initialized with zeros on configuration.

If the INITVAL is specified as a hex string of 16 values, the values should not be greater than 3, since 3 is setting both memory locations to 1. If a value greater than 3 is used for synthesis or mapping, only the last two (least significant) bits are used for initialization by the mapper.For example,

INITVAL=0x00000000ffffffff

is equivalent to

INITVAL=0x0000000033333333

for mapping purposes, since the first two bits of the "f" are ignored.

You can refer to the following technical note on the Lattice web site for more information and usage.

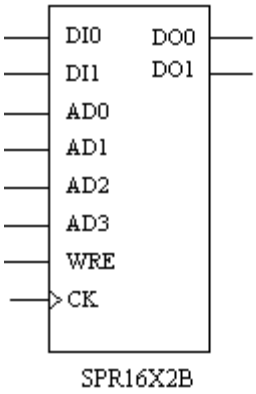
- TN1094 - On-Chip Memory Usage Guide for LatticeSC Devices

SPR16X2B

16 Word by 2 Bit Positive Edge Triggered Write Synchronous Single Port RAM Memory with Positive Write Enable and Positive Write Port Enable (1-Slice)

Architectures Supported:

- LatticeECP/EC
- LatticeXP
- MachXO
- Platform Manager



INPUTS: DI0, DI1, AD0, AD1, AD2, AD3, WRE, CK

OUTPUTS: DO0, DO1

Description

Refer to SPR16X2 for functionality. You can also refer to the following technical notes on the Lattice web site for EBR port definition, attribute definition, and use.

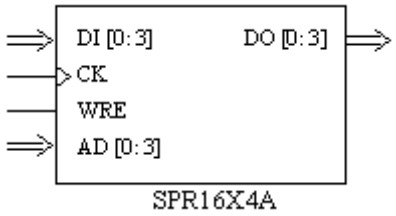
- TN1051 - Memory Usage Guide for LatticeECP/EC and LatticeXP Devices
- TN1092 - MachXO Memory Usage Guide

SPR16X4A

Distributed Single Port RAM

Architectures Supported:

- LatticeECP2/M
- LatticeXP2



INPUTS: DI0, DI1, DI2, DI3, AD0, AD1, AD2, AD3, CK, WRE

OUTPUTS: DO0, DO1, DO2, DO3

Description

PFU based distributed pseudo single port RAM primitive. See Memory Primitives Overview for more information.

You can also refer to the following technical notes on the Lattice web site for port definition, attributes, and use.

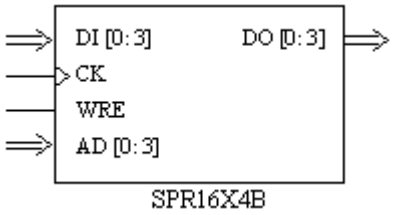
- TN1104 - LatticeECP2/M Memory Usage Guide
- TN1137 - LatticeXP2 Memory Usage Guide

SPR16X4B

Distributed Single Port RAM

Architectures Supported:

- LatticeXP2



INPUTS: DI0, DI1, DI2, DI3, AD0, AD1, AD2, AD3, CK, WRE

OUTPUTS: DO0, DO1, DO2, DO3

ATTRIBUTES:

INITVAL: (*Verilog*) "64'hXXXXXXXXXXXXXXXX" (16-bit hex string)
(*VHDL*) "0xXXXXXXXXXXXXXXXX" (16-bit hex string)
Default: all zeros

Description

PFU based distributed pseudo single port RAM primitive. See Memory Primitives Overview for more information.

You can also refer to the following technical note on the Lattice web site for port definition, attributes, and use.

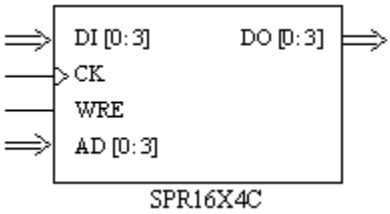
- TN1137 - LatticeXP2 Memory Usage Guide

SPR16X4C

Distributed Single Port RAM

Architectures Supported:

- ECP5
- LatticeECP3
- LFMNX
- LIFMD
- LIFMDF
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUTS: DI3, DI2, DI1, DI0, AD3, AD2, AD1, AD0, CK, WRE

OUTPUTS: DO3, DO2, DO1, DO0

ATTRIBUTES:

INITVAL: "0xXXXXXXXXXXXXXXXX" (16-bit hex string) (default: all zeros)

Description

PFU based distributed Single Port RAM primitive. See Memory Primitives Overview for individual port description.

You can also refer to the following technical notes on the Lattice web site for port definition, attribute definition, and use.

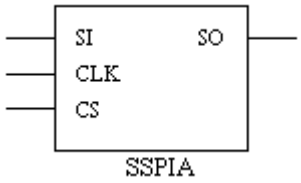
- TN1201 - Memory Usage Guide for MachXO2 Devices
- TN1179 - LatticeECP3 Memory Usage Guide

SSPIA

SSPI TAG Memory

Architectures Supported:

- LatticeXP2



INPUTS: SI, CLK, CS

OUTPUT: SO

ATTRIBUTES:

TAG_INITSIZE: 448, 632, 768, 2184 (default), 2488, 2640, 3384, 3608

TAG_INITIALIZATION: "DISABLED" (default), "ENABLED"

TAG_INITVAL_00 to TAG_INITVAL_0C: "0xXXX...X" (80-bit hex string) (default: all zeros)

Description

Implements the dedicated “TAG Memory” block, which is a one page FLASH non-volatile memory accessible by the hardwired Serial Peripheral Interface port or the JTAG port. This stand-alone TAG memory is ideal for scratch pad memory for mission critical data, board serialization, board revision log and programmed pattern identification.

The LatticeXP2 family of devices provides dedicated TAG memory ranging from 632 to 3384 bits depending on device density. The user can read and write to the TAG memory either through the SPI port (External Slave SPI) or the CIB interface (Internal Slave SPI) by setting the SLAVE_SPI_PORT

attribute. The software default for access to the Tag memory is the CIB interface. The TAG memory initialization file format is similar to that of EBR.

- The TAG interface through the SPI port (External Slave SPI): When the TAG interface is set for the SPI port, four SPI pins will be reserved for the TAG access through the SPI port.
- The TAG interface through the CIB interface (Internal Slave SPI): When the TAG interface is set for the CIB interface, the user can select any four general purpose IO pins to access the TAG memory using the SPI commands. The four SPI pins on each LatticeXP2 device are considered general purpose IOs if they are not reserved using SLAVE_SPI_PORT attribute.

SSPIA Port Description

Table 885:

Port Name	I/O	Description
SI	I	Data input
CLK	I	Clock
CS	I	Chip select
SO	O	Data output

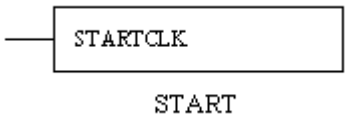
You can refer to LatticeXP2 technical notes on the Lattice web site for more details.

START

Startup Controller

Architectures Supported:

- ECP5
- LatticeECP3
- LatticeXP2
- LFMNX
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager 2



INPUT: STARTCLK

Description

This primitive determines the user clock for the Wake up sequence. You can instantiate this module in your HDL source to tie a specific user clock to be used in the wake-up sequence instead of the TCK (JTAG), BCLK (SDM), or MCLK/CCLK (sysCONFIG).

START Usage with Verilog HDL

```
module START (STARTCLK);
input  STARTCLK;
endmodule
```

START Usage with VHDL

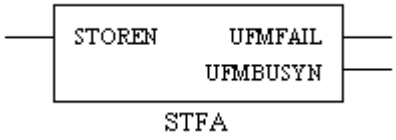
```
COMPONENT START
   PORT(
        STARTCLK    :  IN STD_ULOGIC
       );
END COMPONENT;
```

STFA

Store to Flash Primitive

Architectures Supported:

- LatticeXP2



INPUT: STOREN

OUTPUTS: UFMFAIL, UFMBUSYN

Description

The LatticeXP2 store-to-flash primitive is for user flash module (UFM) operations. The main function of a UFM is to protect the user data from being lost when the system is powered OFF. The user data in the UFM will be used by the system for initialization when the power comes back. To emulate the

UFM capability, the users can use the EBR (shadow flash) memory configurations and then transfer the data to UFM through Store-to-Flash operation. The store-to-flash operation is a single-command-two-operation process. When the store-to-flash operation is initiated, an erase-UFM-Flash CIB signal will be enabled to erase the Flash, followed by the transfer-to-flash operation. Once the transfer is done, the flash controller will send a transfer-done signal back to the user logic. During the Store-to-Flash operation, the EBR's are not accessible. There is no difference between regular EBR RAM configuration and shadow flash (UFM) EBR RAM configuration in Lattice Diamond GUI. The presence of a STFA primitive in the design determines EBR RAM configuration. Due to a silicon limitation, the user cannot use the Store-to-Flash operation if the SED is operating in an Always mode. Only one STFA instance in the design is allowed.

STFA Port Description

Table 886:

Port Name	Corresponding Hardware Port Name	I/O	Description
STOREN	storecmdn	I	Initiates to store the EBR content to Flash
UFMFAIL	ufm_fail	O	Store to Flash operation failed
UFMBUSYN	fl_busyn	O	Tells the user whether the FLASH is in busy state or not

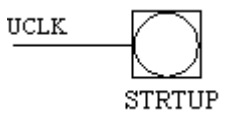
You can refer to LatticeXP2 technical notes on the Lattice web site for more information.

STRTUP

Startup Controller

Architectures Supported:

- LatticeECP/EC
- LatticeECP2/M
- LatticeSC/M



INPUT: UCLK

Description

After configuration, the FPGA enters the start-up phase, which is the transition between the configuration and operational states. Normally, the relative timing of the following three events is triggered by the configuration clock (CCLK): DONE going high, release of the set/reset of internal FFs, and activation of user I/Os. The three events can also be triggered by a user clock, UCLK. This allows the start-up to be synchronized by a known system clock. For more detailed information refer to an available data book or contact technical support.

Another set of bitstream options for the STRTUP block allows the DONE pin to be held low and then released to be used with either CCLK or UCLK to control the release of the set/reset of internal FFs and the activation of user I/Os. This allows the synchronization of the start-up of multiple FPGAs.

UCLK: User defined clock to trigger DONE going high, release of set/reset of internal FFs, and activation of user I/Os.

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in Schematic Editor.

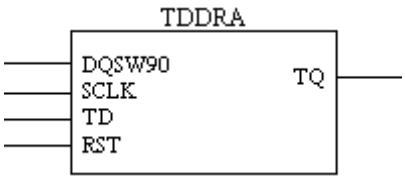
T

TDDRA

Tristate for DQ/DQS of PIC Cell

Architectures Supported:

- MachXO2
- Platform Manager 2



INPUTS: DQSW90, SCLK, TD, RST

OUTPUT: TQ

ATTRIBUTES:

GSR: "ENABLED" (default), "DISABLED"

DQSW90_INVERT: "DISABLED" (default), "ENABLED"

Description

TDDRA is the tristate for DQ/DQS of the PIC cell. It is used for right side only. See the below table for the port description.

Table 887:

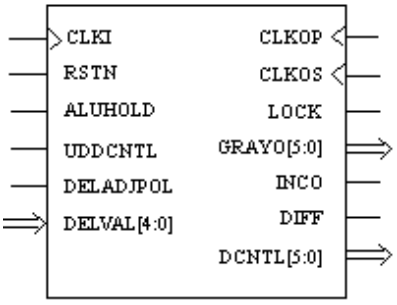
Signal	I/O	Description
DQSW90	I	Shifts the DQS signal by 90 degree
SCLK	I	Clock from the CIB
TD	I	Tristate signal
RST	I	RESET to this block from the CIB
TQ	O	Tristate output for DQ

TR1DLLB

Time Reference DLL with Dynamic Delay Adjustment

Architectures Supported:

- LatticeECP3



TR1DLLB

INPUTS: CLKI, RSTN, ALUHOLD, UDDCNTL, DELADJPOL, DELVAL4, DELVAL3, DELVAL2, DELVAL1, DELVAL0

OUTPUTS: CLKOP, CLKOS, LOCK, INCO, DIFF, GRAYO5, GRAYO4, GRAYO3, GRAYO2, GRAYO1, GRAYO0, DCNTL5, DCNTL4, DCNTL3, DCNTL2, DCNTL1, DCNTL0;

ATTRIBUTES:

CLKOP_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_FPHASE: 0, 11, 22, 33, 45, 56, 67, 78, 90, 101 (default), 112, 123, 135, 146, 157, 169, 191, 202, 214, 225, 236, 247, 259, 281, 292, 304, 315, 326, 337, 349

CLKOS_DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

CLKOS_FPHASE _ADJVAL: integers -20~20 (default: 0)

ALU_LOCK_CNT: integers 3~15 (default: 3)

ALU_UNLOCK_CNT: integers 3~15 (default: 3)

GLITCH_TOLERANCE: integers 0~7 (default: 2)

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

CLKOP_DUTY50: "DISABLED" (default), "ENABLED"

CLKOS_DUTY50: "DISABLED" (default), "ENABLED"

Description

TRDLLB specifies the Time Reference operation mode for the general purpose DLL (GDLL). It supports the 1G SPI4.2 interface. The feedback connection is not required for this mode and hence the CLKFB is not captured on the TRDLLB primitive. In this mode, the CLKFB should be tied to GND.

Port Description

Table 888:

Port Name	Optional	Logical Capture Port Name
ALUHOLD	YES	HOLD
RSTN	YES	RSTN
UDDCNTL	YES	UDDCNTL
CLKI	NO	CLKI
CLKOP	YES	CLKOP
CLKOS	YES	CLKOS
LOCK	NO	LOCK
GRAYO[5:0]	YES	GRAY_OUT[5:0]
INCO	YES	INC_OUT
DIFF	YES	DIFF
DCNTL[5:0]	NO	DCNTL[5:0]
DELADJPOL	YES	DELADJPOL
DELVAL[4:0]	YES	DELVAL[4:0]

For more information, refer to the following technical note on the Lattice web site:

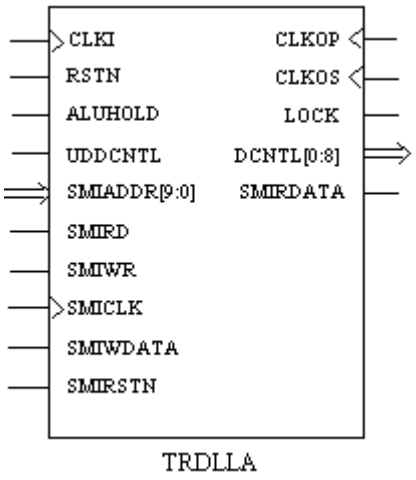
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide

TRDLLA

Time Reference Delay

Architectures Supported:

- LatticeECP2/M
- LatticeSC/M



INPUTS: CLKI, RSTN, ALUHOLD, UDDCNTL, SMIADDR9, SMIADDR8, SMIADDR7, SMIADDR6, SMIADDR5, SMIADDR4, SMIADDR3, SMIADDR2, SMIADDR1, SMIADDR0, SMIRD, SMIWR, SMICLK, SMIWDATA, SMIRSTN

OUTPUTS: CLKOP, CLKOS, LOCK, DCNTL0, DCNTL1, DCNTL2, DCNTL3, DCNTL4, DCNTL5, DCNTL6, DCNTL7, DCNTL8, SMIRDATA

ATTRIBUTES:

CLKOP_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_PHASE: (0, 90, 180, 270, 360) + (0, 11, 22, 45) (default: 0)

CLKOS_FPHASE: 0 (default), 11, 22, 45

CLKOP_DUTY50: "DISABLED" (default), "ENABLED"

CLKOS_DUTY50: "DISABLED" (default), "ENABLED"

CLKOS_DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

CLKOS_FDEL_ADJ: "DISABLED" (default), "ENABLED"

CLKOS_FPHASE _ADJVAL: integers -127~127 (default: 0)

ALU_LOCK_CNT: integers 3~15 (default: 3)

ALU_UNLOCK_CNT: integers 3~15 (default: 3)

GLITCH_TOLERANCE: integers 0~7 (default: 2 for LatticeECP2/M, 0 for LatticeSC/M)

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

(*LatticeSC/M only*) DCNTL_ADJVAL: integers -127~127 (default: 0)

(*LatticeSC/M only*) SMI_OFFSET: 0x400~0x7FF (default: 12'h410)

(*LatticeSC/M only*) MODULE_TYPE: "TRDLLA"

(*LatticeSC/M only*) IP_TYPE: "TRDLLA"

Description

TRDLLA will generate four phases of the clock, 0, 90, 180, 270 degrees, along with the control setting used to generate these phases. This mode features registered control bit output with separate enable, addition and subtraction on the outgoing control bits, lock achieved starting from minimum delay which guarantees lock to first harmonic (fundamental frequency), and four available output phases (0, 90, 180, 270) degrees. This requires internal feedback only, a maximum frequency 700MHz, and a minimum frequency 100MHz.

For more information, see the following technical notes on the Lattice web site:

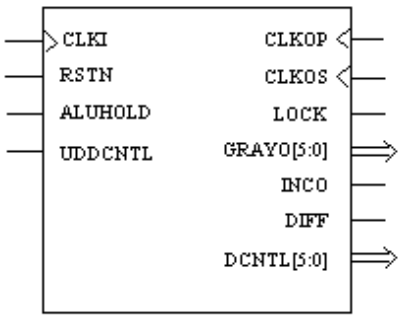
- TN1098 - LatticeSC sysCLOCK and PLL/DLL User's Guide
- TN1103 - LatticeECP2 sysCLOCK PLL Design and Usage Guide

TRDLLB

Time Reference DLL

Architectures Supported:

- LatticeECP3



TRDLLB

INPUTS: CLKI, RSTN, ALUHOLD, UDDCNTL

OUTPUTS: CLKOP, CLKOS, LOCK, INCO, DIFF, GRAYO5, GRAYO4, GRAYO3, GRAYO2, GRAYO1, GRAYO0, DCNTL5, DCNTL4, DCNTL3, DCNTL2, DCNTL1, DCNTL0

ATTRIBUTES:

CLKOP_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_PHASE: 0 (default), 90, 180, 270, 360

CLKOS_FPHASE: 0, 11, 22, 33, 45, 56, 67, 78, 90, 101 (default), 112, 123, 135, 146, 157, 169, 191, 202, 214, 225, 236, 247, 259, 281, 292, 304, 315, 326, 337, 349

CLKOS_DIV: 1 (default), 2, 4

GSR: "DISABLED" (default), "ENABLED"

CLKOS_FPHASE _ADJVAL: integers -20~20 (default: 0)

ALU_LOCK_CNT: integers 3~15 (default: 3)

ALU_UNLOCK_CNT: integers 3~15 (default: 3)

GLITCH_TOLERANCE: integers 0~7 (default: 2)

LOCK_DELAY: integers 0~1000 (in ns) (default: 100)

CLKOP_DUTY50: "DISABLED" (default), "ENABLED"

CLKOS_DUTY50: "DISABLED" (default), "ENABLED"

Description

TRDLLB specifies the Time Reference operation mode for the general purpose DLL (GDLL). The feedback connection is not required for this mode and hence the CLKFB is not captured on the TRDLLB primitive. In this mode, the CLKFB should be tied to GND.

Port Description

Table 889:

Port Name	Optional	Logical Capture Port Name
ALUHOLD	YES	HOLD
RSTN	YES	RSTN
UDDCNTL	YES	UDDCNTL
CLKI	NO	CLKI
CLKOP	YES	CLKOP
CLKOS	YES	CLKOS
LOCK	NO	LOCK
GRAYO[5:0]	YES	GRAY_OUT[5:0]
INCO	YES	INC_OUT
DIFF	YES	DIFF
DCNTL[5:0]	NO	DCNTL[5:0]

For more information, refer to the following technical note on the Lattice web site:

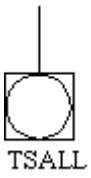
- TN1178 - LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide

TSALL

Global Tristate Interface

Architectures Supported:

- LatticeSC/M
- LFMNX
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUT: TSALL (TSALLN for LatticeSC/M)

Description

TSALL is used to tristate buffers in your design. The TSALL component is connected to a net to drive all output and bidirectional buffers into a HIGH impedance state when active HIGH.

It is not necessary to connect signals to buffers explicitly. The function will be implicitly connected globally.

Note

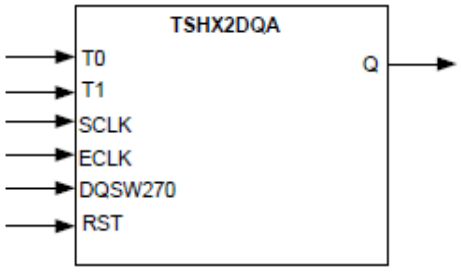
- The TSALL component may be driven by general FPGA logic or by the read-configuration block. In the latter case, the TSALL block must be driven by a buffer located at the RDCGFN pin. When locating the TSALL to the RDCFGN, you must do this by explicitly designating “RDCFGN” in the attribute. Check with customer support or with FAEs for more details.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

TSHX2DQA

Generates the tristate control for DQ data output for DDR2 and DDR3 memory

Architectures Supported:

- ECP5



INPUTS: T0, T1, SCLK, DQSW270, RST

OUTPUT: Q

Description

This primitive is used to generate the tristate control for DQ data output for DDR2 and DDR3 memory.

Table 890:

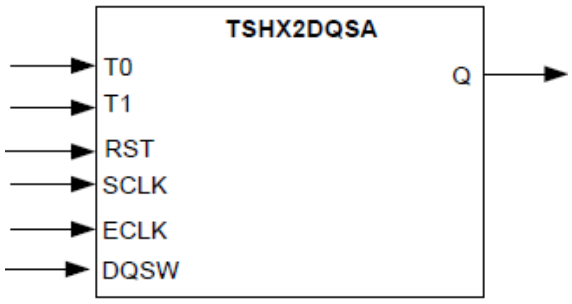
Signal	I/O	Description
T0, T1	I	Tristate input
ECLK	I	ECLK input (2x speed of SCLK)
DQSW270	I	Clock that is 90 degree ahead of clock used to generate the DQS output
SCLK	I	SLCK input
RST	I	Reset input
Q	O	Tristate output

TSHX2DQSA

Generates the tristate control for DQS output

Architectures Supported:

- ECP5



INPUTS: T0, T1, SCLK, DQSW, ECLK, RST

OUTPUT: Q

Description

This primitive is used to generate the tristate control for DQS output.

Table 891:

Signal	I/O	Description
T0, T1	I	Tristate input
ECLK	I	ECLK input (2x speed of SCLK)

Table 891:

Signal	I/O	Description
DQSW	I	DQSW includes write leveling phase shift from ECLK
SCLK	I	SLCK input
RST	I	Reset input
Q	O	Tristate output

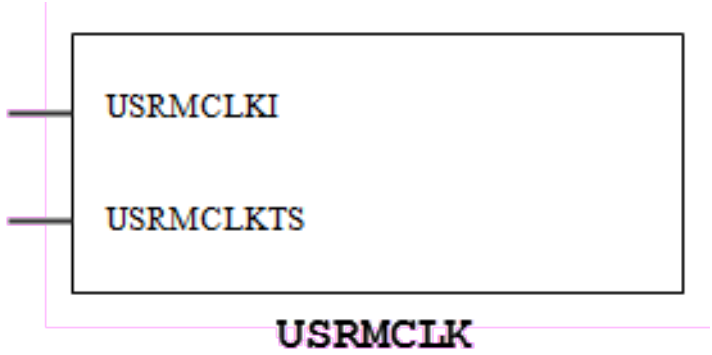
U

USRMCLK

Allows User Function to Access SPI PROM

Architectures Supported:

- ECP5



INPUTS: USRMCLKI, USRMCLKTS

Description

A primitive to allow the user function to access the SPI PROM. It has two inputs. This feature allows the user to tie a specific user clock to be used as MCLK. Without this instantiation the device will use the CFG MCLK as MCLK. This primitive can only be instantiated if the device is in MSPI mode. DRC error will be issued if this primitive be instantiated in any mode other than MSPI. The table below describes the ipnputs of the USRMCLK primitive.

Table 892:

Port	I/O	Function
USRMCLKI	I	User defined MCLK.
USRMCLKTS	I	User defined MCLK tri-state.

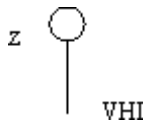
V

VHI

Logic High Generator

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



OUTPUT: Z

Note

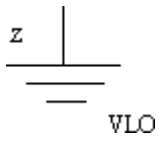
- It is possible that this primitive will be optimized by the back-end tool before place and route.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

VLO

Logic Low Generator

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- Platform Manager
- Platform Manager 2



OUTPUT: Z

Note

- It is possible that this primitive will be optimized by the back-end tool before place and route.
- This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

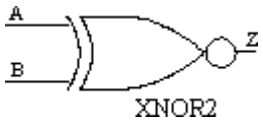
X

XNOR2

2-Input Exclusive NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B

OUTPUT: Z

Note

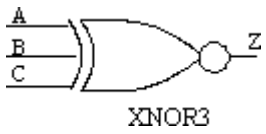
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

XNOR3

3-Input Exclusive NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C

OUTPUT: Z

Note

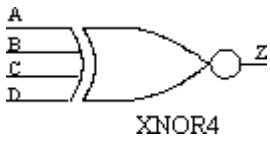
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

XNOR4

4-Input Exclusive NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

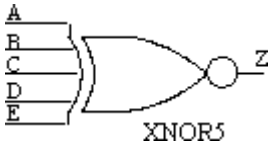
XNOR5

5-Input Exclusive NOR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3

- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

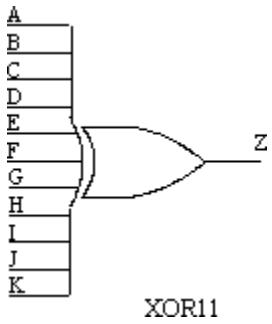
XOR11

11-Input Exclusive OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX

- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E, F, G, H, I, J, K

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

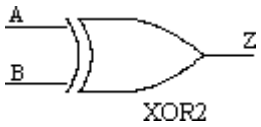
XOR2

2-Input Exclusive OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX

- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

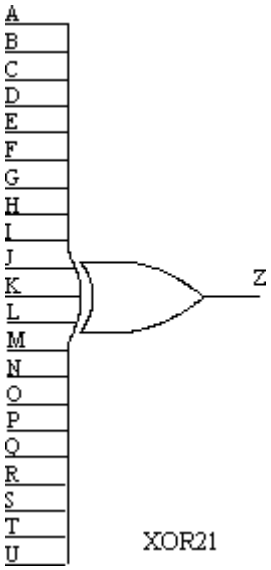
XOR21

21-Input Exclusive OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D

- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

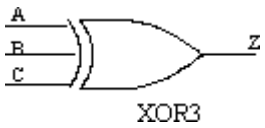
XOR3

3-Input Exclusive OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX

- LIFMD
- LIFMDF
- MachXO
- MachXO2
- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

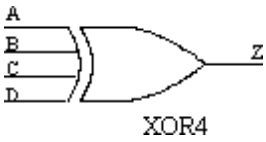
XOR4

4-Input Exclusive OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LFMNX
- LIFMD
- LIFMDF
- MachXO
- MachXO2

- MachXO3D
- MachXO3L
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D

OUTPUT: Z

Note

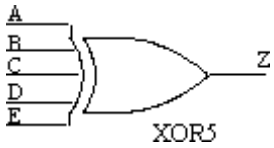
This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

XOR5

5-Input Exclusive OR Gate

Architectures Supported:

- ECP5
- LatticeECP/EC
- LatticeECP2/M
- LatticeECP3
- LatticeSC/M
- LatticeXP
- LatticeXP2
- LIFMD
- LIFMDF
- MachXO
- MachXO2
- Platform Manager
- Platform Manager 2



INPUTS: A, B, C, D, E

OUTPUT: Z

Note

This primitive is also available as a schematic symbol. You can add it to your schematic using the Add > Symbol command in the Schematic Editor.

Primitive-Specific HDL Attributes

The following is a comprehensive list of HDL attributes that are commonly set automatically during module generation and are associated with specific primitives. We list these attributes here mainly for purposes of identification.

In almost all cases, it is not recommended that any of these attributes be edited manually. They should appear in your source as a result of module generation in IPexpress only. There are many interdependencies that exist between certain related attributes and their valid values on an architecture or device basis. These interdependencies make it impractical to simply edit the source HDL. If you were to do so, its very likely an invalid value may result in a failure in your design.

List of Primitive-Specific HDL Attributes

The below table lists all the primitive-specific attributes in alphabetic order.

Table 893:

Attribute	Type	Allowed Values	Default	Description
AEPOINTER	Binary	(*LatticeSCM*) 15-bit binary value; (*MachXO/Platform Manager*) 14-bit binary value	All zeros	Specifies the Almost Empty Flag Pointer. AEPOINTER1 refers to the Almost Empty Flag Pointer 1.
AFPOINTER	Binary	(*LatticeSCM*) 15-bit binary value; (*MachXO/Platform Manager*) 14-bit binary value	All zeros	Specifies the Almost Full Flag Pointer. AFPOINTER1 refers to the Almost Full Flag Pointer 1.
ALU_INIT_CNTVAL	Integer	(*LatticeECP3*) 0 to 31; (*Others*) 0, 4, 8, 12, 16, 32, 48, 64, 72 (0 = DISABLED)	0	Specifies the minimum number of delay taps that ALU will count for. This forces the ALU to count for a minimum number of delay taps before it can find lock, and prevents the DLL finding lock at the minimum possible delay setting and then “falling off the end” of the delay chain if the input clock has jitter. Used in all clock injection cancellation modes where the lock point is not predictable.
ALU_LOCK_CNT	Integer	3 to 15	3	Specifies the Lock Count Cycles. Attached to a DLL-type primitive.
ALU_UNLOCK_CNT	Integer	3 to 15	3	Specifies the Unlock Count Cycles. Attached to a DLL-type primitive.
ASYNC_RESET_RELEASE	String	SYNC, ASYNC	SYNC	Specifies reset release when the reset mode is ASYNC.

Table 893:

Attribute	Type	Allowed Values	Default	Description
BANKID	Integer	0, 1, 2, 3, 4, 5	0	Specifies the ID of the bank that enables dynamic InRD control for the BCINRD primitive.
BGOFF	Boolean	TRUE, FALSE	FALSE	Turns on or off Bandgap when in standby.
BOOT_OPTION	String	INTERNAL, EXTERNAL	INTERNAL	Specifies device boot from external SPI flash or internal flash.
CAS_MATCH_REG	Boolean	TRUE, FALSE	FALSE	Specifies the Cascade Match Register option. Attached to DSP primitives such as MULT9X9C and MULT18X18C.
CFGx_INIT_PAGES	Integer	0 to Max Number of Pages in UFM array.	0	Specifies the number of pages with initialization data, where x holds value of 0 or 1. If the user doesn't enter a value, it is set to 0.
CFGx_INIT_START_PAGE	Integer	Calculated by Software.	0	Software calculates and displays the starting page of the init data in the CFG array, where x holds value between 0 to 1. Read only field.
CFGx_INIT_ALL_ZEROS	Boolean	ENABLED, DISABLED	ENABLED	Specifies that the CFG is initialized with all 0s. x can hold a value between 0 to 1. By default, the radio button is selected.
CFGx_INIT_FILE_NAME	String	String	None	Uploads the CFG Initialization data file, where x holds value between 0 to 1. By default, the radio button is not selected. The browse button is not enabled if the radio button is not selected.
CFGx_INIT_FILE_FORMAT	String	HEX, BIN	HEX	Specifies the selection of file format, either Binary or Hexadecimal, x holds a value between 0 and 1.
CHECKALWAYS	Boolean	ENABLED, DISABLED	DISABLED	When set to ENABLED, makes the SED (Soft Error Detect) run automatically every time upon power up and after device configuration. The software will set signals from the SED IP that puts it in an "Always Running" state.
CLKFB_DIV	Integer	Vary	1	Specifies the CLKFB N Divider setting. Attached to a PLL-type primitive (such as EHXPLLB).
CLKFB_FDEL	Integer	0, 100, 200, ..., 700	0	Specifies the CLKFB Fine Delay setting for the EHXPLLA primitive.

Table 893:

Attribute	Type	Allowed Values	Default	Description
CLKI_DIV	Integer	Vary	1	Specifies the CLKI M Divider setting. Attached to a PLL or DLL primitive (such as EHXPLLB and CIDDLLA).
CLKI_FDEL	Integer	0, 100, 200, ..., 700	0	Specifies the CLKI Fine Delay setting for the EHXPLLA primitive.
CLKMODE	String	ECLK, SCLK	ECLK	Specifies the edge clock or system clock for the CLKCNTL primitive.
CLKOK_BYPASS	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables the GPLL clock bypass feature. When enabled, this feature allows the input reference clock (CLK) to bypass the PLL and directly drive CLKOP, CLKOS, and CLKOK. If CLKOP is used for bypass, the PLL is no longer functional and cannot be used as a PLL. The CLKOS and CLKOK can be used for bypass without affecting the operation of the loop. IPexpress includes selections for CLKOP, CLKOS, and CLKOK bypass.
CLKOK_DIV	Integer	Even integers from 2 to 128	2	Specifies the CLKOK K Divider setting. Attached to a PLL-type primitive (such as EHXPLLB).
CLKOK_INPUT	String	CLKOP, CLKOS	CLKOP	Specifies the CLKOK divider input. Attached to a PLL-type primitive. (such as EHXPLLF).
CLKOP_BYPASS	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables the GPLL clock bypass feature. When enabled, this feature allows the input reference clock (CLK) to bypass the PLL and directly drive CLKOP, CLKOS, and CLKOK. If CLKOP is used for bypass, the PLL is no longer functional and cannot be used as a PLL. The CLKOS and CLKOK can be used for bypass without affecting the operation of the loop. IPexpress includes selections for CLKOP, CLKOS, and CLKOK bypass.
CLKOP_DIV	Integer	Vary	1 or 8	Specifies the CLKOP V Divider setting. Attached to a PLL-type primitive (such as EHXPLLB). **Note**: CLKOP_DIV value must be calculated to maximize the FVCO within the specified range based on CLKI_DIV and CLKFB_DIV values for optimum performance.

Table 893:

Attribute	Type	Allowed Values	Default	Description
CLKOP_DUTY50	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables the CLKOP Duty Cycle. Attached to a PLL-type primitive (such as EHXPLLA).
CLKOP_MODE	String	BYPASS, FDEL0, VCO, DIV	BYPASS	Specifies the CLKOP Select for the EHXPLLA primitive.
CLKOP_PHASE	Integer	0, 90, 180, 270, 360	0	Specifies the CLKOP Phase setting. Attached to a DLL-type primitive (such as TRDLLA).
CLKOP_TRIM_DELAY	Integer	0 to 7	0	Specifies the CLKOP Duty Trim Polarity Delay. Attached to a PLL-type primitive.
CLKOP_TRIM_POL	String	FALLING, RISING	FALLING for LatticeXP2; RISING for LatticeECP 3	Specifies the CLKOP Duty Trim Polarity. Attached to a PLL-type primitive.
CLKOS_BYPASS	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables the GPLL clock bypass feature. When enabled, this feature allows the input reference clock (CLK) to bypass the PLL and directly drive CLKOP, CLKOS, and CLKOK. If CLKOP is used for bypass, the PLL is no longer functional and cannot be used as a PLL. The CLKOS and CLKOK can be used for bypass without affecting the operation of the loop. IPexpress includes selections for CLKOP, CLKOS, and CLKOK bypass.
CLKOS_DIV	Integer	1, 2, 4 for DLL primitives; 1 to 64 for PLL primitives	1	Specifies the CLKOS Divider setting. Attached to a PLL- or DLL-type primitive (such as EHXPLLA and CIDDLLA).
CLKOS_DUTY50	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables the CLKOS Duty Cycle. Attached to a PLL-type primitive (such as EHXPLLA).
CLKOS_FDEL	Integer	0, 100, 200, ..., 700	0	Specifies the CLKOS Fine Delay setting for the EHXPLLA primitive.
CLKOS_FDEL_ADJ	Boolean	ENABLED, DISABLED	DISABLED	Specifies the CLKOS DEL Manual Setting Adjust Value. Attached to a DLL-type primitive (such as TRDLLA).
CLKOS_FPHASE	Integer	Vary	0	Specifies the CLKOS Fine Phase setting. Attached to a DLL-type primitive (such as TRDLLA).

Table 893:

Attribute	Type	Allowed Values	Default	Description
CLKOS_FPHASE _ADJVAL	Integer	-20 to 20	0	Specifies the CLKOS Fine Phase Adjust Value. Attached to a DLL-type primitive (such as TRDLLA).
CLKOS_MODE	String	BYPASS, FDEL, VCO, DIV	BYPASS	Specifies the CLKOS Select for the EHXPLLA primitive.
CLKOS_PHASE	Integer	Vary	0	Specifies the CLKOS Phase setting. Attached to a DLL-type primitive (such as TRDLLA).
CLKOS_TRIM_DELAY	Integer	0 to 3	0	Specifies the CLKOS Duty Trim Polarity Delay. Attached to a PLL-type primitive.
CLKOS_TRIM_POL	String	RISING, FALLING	RISING	Specifies the CLKOS Duty Trim Polarity Delay. Attached to a PLL-type primitive.
CLKOS_VCODEL	Integer	0 to 31	0	Specifies the CLKOS VCO Delay setting for the EHXPLLA primitive.
CRUDIV	Integer	1.0, 2.0, 3.5, 4.0, 5.0	5.0	Sets the divider setting for the DIVCLK output.
CSDECODE	Binary	2- or 3-bit binary value	Vary	Attached to a single-port block RAM primitive. The CSDECODE value determines the decoding value of CS[2:0]. A value set to “000” means that the memory is selected if CS[2:0]=1’B000.
DATA_WIDTH	Integer	1, 2, 4, 9, 18 for DATA_WIDTH, DATA_WIDTH_A, and DATA_WIDTH_B; 1, 2, 4, 9, 18, 36 for DATA_WIDTH_R and DATA_WIDTH_W	9, 18, or 36	Specifies the Data Word Width. Attached to a memory type primitive.
DCNTL_ADJVAL	Integer	-127 to 127	0	Specifies the Adjust Delay Control. Attached to a DLL-type primitive.
DCSMODE	String	NEG, POS, HIGH_LOW, HIGH_HIGH, LOW_LOW, LOW_HIGH, CLK0, CLK1	NEG	Sets the particular mode for the DCS primitive. Refer to DCSMODE Values for more information.
DEL_ADJ	String	PLUS, MINUS	PLUS	Specifies the delay adjustment sign bit.
DEL_VAL	Integer	0 to 127 if DEL_ADJ=PLUS; 1 to 128 if DEL_ADJ=MINUS	0	Specifies the delay adjustment offset.
DEL[0,1,2,3,4]_GRAY	Boolean	ENABLED, DISABLED	DISABLED	Specifies gray in for DEL0, DEL1, DEL2, DEL3, and DEL4. Attached to a DLL-type primitive.

Table 893:

Attribute	Type	Allowed Values	Default	Description
DEL_MODE	String	SCLK_ZEROHOLD, ECLK_ALIGNED, ECLK_CENTERED, ECLK_CENTERED_MIPI, ECLK_CENTERED_SLVS, SCLK_ALIGNED, SCLK_CENTERED, USER_DEFINED	USER_DEFINED	Controls whether the fixed delay value is dependent on a certain interface or user-defined delay value.
DEL_VALUE	String	DELAY0, DELAY1, DELAY2, ..., DELAY31	DELAY0	Specifies user-defined delay value.
DELAY_CNTL	String	DYNAMIC, STATIC	STATIC	Specifies the Delay Control mode. Attached to a PLL-type primitive (such as EHXPLLB). The DYNAMIC mode switches delay control between DYNAMIC and STATIC depending upon the input logic of the DDAMODE pin. In the STATIC mode, delay inputs are ignored.
DELAY_PWD	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables the CLKOS Fine Delay Powerdown. Attached to a PLL-type primitive. When set to ENABLED, the f_fdelay_pwd fuse will be set to HIGH to disable the fine delay circuitry for power saving. When set to DISABLED, the f_fdelay_pwd fuse will be set to LOW to enable the fine delay circuitry.
DELAY_VAL	Integer	0 to 15	0	Specifies the CLKOS Fine Delay Value. Attached to a PLL-type primitive.
DEV_DENSITY	String	Vary	Vary	Specifies the device density.
DIV	Integer	1, 2, 4 (1:off) for the CLKDIV primitive; 2.0, 3.5, 4.0 for the CLKDIVC primitive; 1, 2, 4, 8, 16, 32, 64, 128 for the OSCA primitive	1 or 2 or 2.0	Specifies the Divider setting.
DQS_LI_DEL_ADJ	String	PLUS, MINUS	Vary	Adjusts the sign delay offset direction for input DDR. For DQSBUFH, it adjusts the sign bit for the READ delay.

Table 893:

Attribute	Type	Allowed Values	Default	Description
DQS_LI_DEL_VAL	Integer	0 to 63 or 0 to 127 if DQS_LI_DEL_ADJ=PLUS; 1 to 64 or 1 to 128 if DQS_LI_DEL_ADJ =MINUS	Vary	Specifies the delay value for input DDR.
DQS_LO_DEL_ADJ	String	PLUS, MINUS	PLUS	Adjusts the sign delay offset direction for output DDR. For DQSBUFH, it adjusts the sign bit for the WRITE delay.
DQS_LO_DEL_VAL	Integer	0 to 63 or 0 to 127 if DQS_LO_DEL_ADJ=PLUS; 1 to 64 or 1 to 128 if DQS_LO_DEL_ADJ=MINUS	0	Specifies the delay value for output DDR.
DQSW90_INVERT	String	DISABLED, ENABLED	DISABLED	Selects the clock polarity for the second FF of the tristate cell for the DQS pin. Only used for the DQS during DDR write.
DR_CONFIG	String	DISABLED, ENABLED	DISABLED	Indicates whether the primitive is used for data recovery configuration or not. If it is for data recovery configuration then the correct GBB timing will be set. This attribute is required for mapper and is not required for simulation.
DUTY	Integer	Vary	4 or 8	Specifies the duty cycle floating point percentage value. Used to control the duty cycle modes of PLL primitives such as EHXPLLB.
DYNDEL_CNTL	String	STATIC, DYNAMIC	DYNAMIC	Enables the static or dynamic delay. Attached to a DQSBUF primitive (such as DQSBUFB).
DYNDEL_TYPE	String	NORMAL, SHIFTED	NORMAL	Specifies the value of the Static Delay input to the write portion of the DQSBUFD or DQSBUFE module that controls the clock inversion. NORMAL: 0-degree phase shift; SHIFTED: 180-degree phase shift through clock inversion.
DYNDEL_VAL	Integer	0 to 127	0	Specifies the value of the Static Delay input to the write portion of the DQSBUFD or DQSBUFE module.
EFB_I2C1	Boolean	DISABLED, ENABLED	DISABLED	Enables I2C Primary User Port.
EFB_I2C2	Boolean	DISABLED, ENABLED	DISABLED	Enables I2C Secondary User Port.

Table 893:

Attribute	Type	Allowed Values	Default	Description
EFB_SPI	Boolean	DISABLED, ENABLED	DISABLED	Enables the SPI Port.
EFB_TAMPER_TYPE_PASSWORD	Boolean	DISABLED, ENABLED	DISABLED	Enables the detection of the unauthorized access to password protection.
EFB_TAMPER_TYPE_LOCKED_FLASH_SRAM	Boolean	DISABLED, ENABLED	DISABLED	Enables the action detection of unauthorized access to locked flash sector or SRAM.
EFB_TAMPER_TYPE_MANUFACTURE_MODE	Boolean	DISABLED, ENABLED	DISABLED	Enables the detection of entering manufacture mode.
EFB_TAMPER_SRC_JTAG	Boolean	DISABLED, ENABLED	DISABLED	Enables the threat detection for JTAG port.
EFB_TAMPER_SRC_SSPI	Boolean	DISABLED, ENABLED	DISABLED	Enables the threat detection for SSPI port.
EFB_TAMPER_SRC_SI2C	Boolean	DISABLED, ENABLED	DISABLED	Enables the threat detection for SI2C port.
EFB_TAMPER_SRC_WB	Boolean	DISABLED, ENABLED	DISABLED	Enables the threat detection for WB port.
EFB_TAMPER_PORT_LOCK	Boolean	DISABLED, ENABLED	DISABLED	Enables the detection of accessing port lock function.
EFB_TAMPER_DETECTION_RESPONSE	Boolean	DISABLED, ENABLED	DISABLED	
EFB_TC	Boolean	DISABLED, ENABLED	DISABLED	Enables Timer/Counter port.
EFB_TC_PORTMODE	String	WB, NO_WB	NO_WB	Determined by T/C Static and Dynamic radio buttons. Selecting “Use static ...” sets it to NO_WB “Dynamic register ...” sets it to WB and enables WB port. It brings the primitive port TCIC to the top level.
EFM_UFM_BOOT	String	INT_SINGLE_BOOT_CFG0 INT_SINGLE_BOOT_CFG0_UFM0 INT_DUAL_BOOT_CFG0_CFG1 INT_DUAL_BOOT_CFG0_UFM0_CFG1 INT_DUAL_BOOT_CFG0_CFG1_UFM1 INT_DUAL_BOOT_CFG0_UFM0_CFG1_UFM1 EXTERNAL_BOOT	INT_SINGLE_BOOT_CFG0	Specifies the UFM boot setting.

Table 893:

Attribute	Type	Allowed Values	Default	Description
EFB_UFM	Boolean	DISABLED, ENABLED	DISABLED	Indicates that the user wants to use one of UFM0, UFM1, UFM2, UFM3, CFG0, or CFG1.
EFB_UFMx	Boolean	DISABLED, ENABLED	DISABLED	Indicates that the user wants to address the UFMx through WB, where x is a value between 0 and 3. It enables WB port and WBCUFMIRQ port.
EFB_CFGx	Boolean	DISABLED, ENABLED	DISABLED	Indicates that the user wants to address the CFGx through WB, where x is a value between 0 and 1. It enables WB port and WBCUFMIRQ port.
EFB_WB_CLK_FREQ	Integer	0 to 133	50	Used by SPI and I2C to derive frequency.
ENCRYPTION	String	ON, OFF	OFF	Specifies the encryption feature.
ER1, ER2	Boolean	ENABLED, DISABLED	ENABLED	Lattice supports two private JTAG instructions ER1 (0x32) and ER2 (0x38). If the ER1 instruction is shifted into the JTAG instruction register, JRTI1 will go high when the TAP controller is in the Run-Test/Idle state. If the ER2 instruction is shifted into the JTAG instruction register, JRTI2 will go high when the TAP controller is in the Run-Test/Idle state.
FB_MODE	String	INTERNAL, CLOCKTREE, EXTERNAL	CLOCKTREE	Defines PLL clock resources in the feedback mode.
FDEL	Integer	-8 to 8	0	Specifies the fine delay adjust setting. Attached to a PLL-type primitive (such as EHXPLLB).
FIN	Real	Vary (in MHz)	100.0	Specifies the input frequency (MHz) designation for a PLL/DLL primitive (such as EHXPLLB, DQSDLLC).
FORCE_MAX_DELAY	Boolean	YES, NO	NO	Used to bypasses the DLL-locking procedure at low frequency. When FIN ≤ 30 MHz (pending PDE result), the software sets this attribute to YES. Then DQSDLL will not go through the locking procedure but will be locked to the maximum delay steps.
FORCE_ZERO_BARREL_SHIFT	Boolean	ENABLED, DISABLED	DISABLED	When set to ENABLED, forces zeros to 18 MSB of shift for barrel shift. Attached to the ALU54A primitive.
FWFT	Boolean	ENABLED, DISABLED	DISABLED	First word fall through.

Table 893:

Attribute	Type	Allowed Values	Default	Description
FULLPOINTER	Binary	(*LatticeSC/M*) 15-bit binary value; (*MachXO/Platform Manager*) 14-bit binary value	All zeros	Specifies the Full Flag Pointer. Attached to a FIFO primitive (such as FIFO8KA). FULLPOINTER1 refers to the Full Flag Pointer 1.
GEARING_MODE	String	X2, X4	X2	Sets gearing mode required.
GLITCH_TOLERANCE	Integer	0 to 7	2	Specifies the Programmable Glitch Tolerance. Attached to a DLL-type primitive.
GSR	Boolean	ENABLED, DISABLED	Vary	Enables or disables the Global Set/ Reset (GSR) for all registered primitives. Applicable to registers, PLLs, and memories such as SP8KA, DP8KA and the like.
INIT	Hexadecimal	Hex value or string	All zeros	Initializes the look-up table values. INIT is required to specify the look-up table values for the LUT primitives (ORCALUT4, 5, 6, 7, or 8). See ORCALUT4 for more information on INIT attribute usage.
INIT_DATA	String	STATIC, DYNAMIC	STATIC	Defines whether or not the memory file can be updated. STATIC – Memory values are stored in the User Flash Memory (UFM) or bitstream but can be shared and can not be updated. DYNAMIC– Memory values are stored in the UFM and can be updated by user logic knowing the EBR address locations.
INITVAL	Hexadecimal	Hex value or string	All zeros	Specifies the initialization value for RAM type primitives (such as SPR16X2 and DPR16X2). These primitives carry prescribed initialization values, for example, DPR16X2 has an initialization value of 0x0000000000000000.
INJECT	Boolean	YES, NO	YES	This injection attribute is not a user selection. It is for software use only. Attached to a Carry Chain primitive.
IP_TYPE	String	EHXPLLA, CIDDLLA, CIMDLLA, TRDLLA, SDCDLLA	Vary	This attribute is not a user selection. It is for software use only. Attached to a PLL- or DLL-type primitive.

Table 893:

Attribute	Type	Allowed Values	Default	Description
I2Cx_ADDRESSING	String	7BIT, 10BIT	7BIT	Specifies the addressing scheme for i2c module: selects between 7-bit addressing and 10-bit addressing in i2c module. The value x represents an index: index 1 is for primary i2c module while index 2 is for the secondary i2c module
I2Cx_SLAVE_ADDR	Binary	Any 7-bit binary value, Any 10-bit binary value	0b0001000 001 (for primary module) 0b0001000 010 (for secondary module)	Specifies a default slave address during i2c hard IP generation. The last 2 bits of the 7-bit addressing and 10-bit addressing are hard-coded. ▶ Address ends xxxxx01 or xxxxxxxx01 for Primary module in user mode. ▶ Address ends xxxxx10 or xxxxxxxx10 for Secondary module.
I2Cx_BUS_PERF		50 kHz, 100kHz, 400 kHz	100 kHz	Master only. No software DRC for Master/Slave mode.
I2Cx_CLK_DIVIDER	Integer	1 to 1024	1	Read-Only display of I2Cx_CLK_DIVIDER value.
I2Cx_GEN_CALL	Boolean	ENABLED, DISABLED	DISABLED	Enables the generic call response in Slave. The attribute is disabled by default.
I2Cx_WAKEUP	Boolean	ENABLED, DISABLED	DISABLED	Enables/disables the i2c core to send a wake up signal to the on chip power manager to wake up the part from standby/sleep mode when the slave address matches. Default is disabled so that the device is in lower power mode
ISI_CAL	String	BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7	BYPASS	Sets the ISI correction values in the ODDRX2D and ODDRX2DQSA blocks.
JTAG_FLASH_PRGRM	Boolean	ENABLED, DISABLED	ENABLED	When set to ENABLED, enables the use of the ispJTAG interface to program or write to Flash devices. Refer to TN1100 - SPI Serial Flash Programming Using ispJTAG in LatticeSC Devices on the Lattice Web site for more information.
LEGACY	Boolean	ENABLED, DISABLED	DISABLED	This attribute is required to support LatticeECP2 to LatticeECP3 mapping. Attached to the ALU54A primitive.

Table 893:

Attribute	Type	Allowed Values	Default	Description
LOCK_CYC	Integer	Integer value	2	This lock cycle attribute is not a user selection. It is for software use only. Attached to a PLL- or DLL-type primitive.
LOCK_DELAY	Integer	0 to 1000 (in ns)	100	This is a PLL-lock time attribute used for simulation. If you wish to enter other than the default value of 100 ns, it can be done by adding this attribute in the DEFPARAM section of the code generated by IPexpress. You can also set this attribute in the Spreadsheet view.
LOCK_SENSITIVITY	String	HIGH, LOW	LOW	This DLL configuration attribute selects greater or less sensitivity to the jitter. **Note**: There is a known issue for the LatticeXP family. The DQSDLL attribute LOCK_SENSITIVITY will always be set to LOW even if you attempt to set it to HIGH. Devices impacted are LFXP20E/C, LFXP15E/C, LFXP10E/C, LFXP6E/C and LFXP3E/C.
LPDDR	String	ENABLED, DISABLED	DISABLED	Turns the LPDDR feature on or off.
LSRMODE	String	EDGE, LOCAL	LOCAL	Attached to DDR and ISR primitives (such as IDDRA and ISRX1A), this attribute takes the EDGE and LOCAL mode options, which allows you to choose Local Set Reset or the Edge Set Reset.
MASK_ADDR	Hexadecimal	Any 4-bit hex value	All zeros	Specifies the starting mask address for the “care bits” mask. Attached to a SED-type primitive.
MASK01	Hexadecimal	Any 14-bit hex value	All zeros	Specifies the mask for EQZM/EQOM. Attached to the ALU54A primitive.
MASKPAT	Hexadecimal	Any 14-bit hex value	All zeros	Specifies the mask for EQPAT/EQPATB. Attached to the ALU54A primitive.
MASKPAT_SOURCE	String	STATIC, DYNAMIC	STATIC	Specifies the EQPAT/EQPATB source setting. Attached to the ALU54A primitive. MASKPAT_SOURCE and MCPAT_SOURCE cannot be DYANMIC at the same time.
MCCLK_FREQ	String	Vary	2.5 or 3.1	Controls the master clock frequency.
MCPAT	Hexadecimal	Any 14-bit hex value	All zeros	Specifies the MEM Cell Pattern. Attached to the ALU54A primitive.

Table 893:

Attribute	Type	Allowed Values	Default	Description
MCPAT_SOURCE	String	STATIC, DYNAMIC	STATIC	Specifies the MEM Cell Pattern source setting. Attached to the ALU54A primitive. MASKPAT_SOURCE and MCPAT_SOURCE cannot be DYANMIC at the same time.
MEMMODE	String	DISABLED, ENABLED	DISABLED	Indicates the memory mode or generic mode.
MODULE_TYPE	String	EHXPLLA, CIDDLLA, CIMDLLA, TRDLLA, SDCDLLA	Vary	This attribute is not a user selection. It is for software use only. Attached to a PLL- or DLL-type primitive.
MULT_BYPASS	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables Multiplier Output Bypass. Attached to DSP primitives such as MULT9X9C and MULT18X18C.
MULT9_MODE	Boolean	ENABLED, DISABLED	DISABLED	Enables or disables the operation in the Mult9 mode. Attached to the ALU54A primitive.
NOM_FREQ	Real	Vary	Vary	Specifies the nominal frequency (in MHz) for oscillator primitives.
NRZMODE	String	DISABLED, ENABLED	DISABLED	Specifies NRZMODE for DDR3_MEM mode for the DQSBUFD primitive.
OSC_DIV	Integer	1, 2, 4, 8, 16, 32, 64, 128, 256	1	Used for the Soft Error Detect (SED). As an attribute for the internal oscillator, OSC_DIV specifies the divisor of the CCLK frequency to be used in the SED module or corresponding primitives.
PHASE_CNTL	String	DYNAMIC, STATIC	STATIC	Specifies the Phase Adjustment Select mode. When this is set to DYNAMIC, the Phase Adjustment Select control switches between Dynamic and Static depending upon the input logic of the DPAMODE pin. If the attribute is set to STATIC, Dynamic Phase Adjustment Select inputs are ignored.
PHASE_DELAY_CNTL	String	DYNAMIC, STATIC	STATIC	Specifies the CLKOS Phase and Duty Control/Duty Trimming mode. Attached to a PLL-type primitive.
PHASE_SHIFT	Integer	45, 57, 68, 79, 90, 101, 112, 123, 135	90	Phase shift value used. This is required for Simulation. DRC Check: This should match the PHASE_SHIFT value set in the DDRDLLA.

Table 893:

Attribute	Type	Allowed Values	Default	Description
PHASEADJ	Real	Vary	0	Specifies the Coarse Phase Shift setting. Attached to a PLL-type primitive (such as EHXPLLB).
PLLCAP	String	ENABLED, DISABLED, AUTO	DISABLED	Enables or disables the external capacitor pin. Attached to a PLL-type primitive. This attribute value will be usedand updated by MPAR. MPAR converts AUTO to DISABLED or ENABLED as per the placement. This attribute has no impact on the simulation.
PLLTYPE	String	AUTO, SPLL, GPLL	AUTO	Specifies the PLL configuration mode. Applicable only for the EPLLD primitive. This attribute value will be used by MPAR. It has no impact on simulation or bit generation.
POROFF	Boolean	TRUE, FALSE	FALSE	Turns on or off POR when in standby.
REG_<*RegisterType*> _<*RegisterName*>	String	Vary	Vary	This attribute applies to DSP block multipliers to enable various registers, such as Input Registers, Pipeline Registers, Output Registers, Signed Registers, Signed Pipeline Registers, and Accumulator Load Pipeline Registers. Attribute value is the register name. For clocks, you can also assign “NONE” to the attribute. Refer to appropriate DSP User Guide on the Lattice Web site for more details.
REGMODE	String	NOREG, OUTREG	NOREG	Specifies the register mode for pipelining.
REGSET	String	SET, RESET	RESET	Sets the output to either SET or RESET for input and output DDR and shift register elements.
RESETMODE	String	ASYNC, SYNC	Vary	Specifies the reset type. This attribute is attached to a block RAM-type primitive that has a memory size smaller than 9 bits. When set to SYNC, the memory reset is synchronized with the clock. When set to ASYNC, the memory reset is asynchronous to the clock.
RNDPAT	Hexadecimal	Any 14-bit hex value	All zeros	Specifies the Rounding Pattern. Attached to the ALU54A primitive.
RST_PULSE	Integer	Integer value	1	Specifies the required reset pulse length. Attached to the Power Up Reset (PUR) primitive.

Table 893:

Attribute	Type	Allowed Values	Default	Description
SCLKLATENCY	Integer	1, 2 (on the top only 1 is valid)	1	Adjusts SCLK latency. For simulation only.
SED_CLK_FREQ	String	2.08, 2.15, 2.22, 2.29, 2.38, 2.46, 2.56, 2.66, 2.77, 2.89, 3.02, 3.17, 3.33, 3.5, 3.69, 3.91, 4.16, 4.29, 4.43, 4.59, 4.75, 4.93, 5.12, 5.32, 5.54, 5.78, 6.05, 6.33, 6.65, 7, 7.39, 7.82, 8.31, 8.58, 8.87, 9.17, 9.5, 9.85, 10.23, 10.64, 11.08, 11.57, 12.09, 12.67, 13.3, 14, 14.78, 15.65, 16.63, 17.73, 19, 20.46, 22.17, 24.18, 26.6, 29.56, 33.25, 38, 44.33, 53.2, 66.5, 88.67, 133	3.5	Specifies the SED clock frequency.
SHIFT_IN	Boolean	TRUE, FALSE	FALSE	Specifies shift for data. Attached to a DSP multiplier primitive.
SMI_OFFSET	Hexadecimal	Hex value	0x410, 12'h410	Specifies Serial Management Interface offset. Attached to a PLL or DLL primitive.
SPI_MODE	String	SLAVE, BOTH	SLAVE	This option allows the user to select between Slave or Both mode for the initial state of the SPI block.
SPI_CLK_DIVIDER	Integer	1 to 64	1	Allows the user to specify a desired master clock frequency. Applies to SLAVE or BOTH operation mode.
SPI_LSB_FIRST	Boolean	ENABLED, DISABLED	DISABLED	Allows for programmable data order (MSB or LSB first) in the SPI core.
SPI_CLK_INV	Boolean	ENABLED, DISABLED	DISABLED	Allows to program the clock polarity used to sample and output data withing the SPI core.
SPI_PHASE_ADJ	Boolean	ENABLED, DISABLED	DISABLED	Allows the user to specify a phase change to match the application. An alternate clock-data relationship is available for SPI devices with particular requirements.
SPI_SLAVE_HANDSHAKE	Boolean	ENABLED, DISABLED	DISABLED	Enables the core to insert certain values into the data stream to inform the external master when data can be sent.

Table 893:

Attribute	Type	Allowed Values	Default	Description
SPI_INTR_TXRDY	Boolean	ENABLED, DISABLED	DISABLED	The core has configurable reset generation for data buffer conditions. The user can specify which of these options is desired.
SPI_INTR_RXRDY	Boolean	ENABLED, DISABLED	DISABLED	The core has configurable reset generation for data buffer conditions. The user can specify which of these options is desired.
SPI_INTR_TXOVR	Boolean	ENABLED, DISABLED	DISABLED	The core has configurable reset generation for data buffer conditions. The user can specify which of these options is desired.
SPI_INTR_RXOVR	Boolean	ENABLED, DISABLED	DISABLED	The core has configurable reset generation for data buffer conditions. The user can specify which of these options is desired.
SPI_WAKEUP	Boolean	ENABLED, DISABLED	DISABLED	Allows the user to enable this feature where the core can optionally provide a wakeup signal to the device to resume from low-power modes. Applies to SLAVE or BOTH operation mode.
STDBYOPT	String	USER, CFG, USER_CFG	USER_CFG	Specifies the entry option for entry signals.
TAG_INITIALIZATION	Boolean	ENABLED, DISABLED	DISABLED	Attached to the SSPIA primitive. When this attribute is set to DISABLED, the TAG configuration will be generated without an initialization file, and TAG_INITVAL_* will be all zeros. When this is set to ENABLED, the TAG configuration will be generated with an initialization file, and TAG_INITVAL_* will contain the initialization data. Any un-initialized byte word will default to 00000000 (software default).
TAG_INITSIZE	Integer	448, 632, 768, 2184, 2488, 2640, 3384, 3608	2184	Specifies the TAG Memory size. Attached to the SSPIA primitive.
TAG_INITVAL	Hexadecimal	Any 80-bit hex value	All zeros	Specifies the TAG initialization value. The 80-bit hex string corresponds to the 320 TAG bits. Attached to the SSPIA primitive.
TC_MODE	String	WATCHDOG, CTCM, FASTPWM, PFCPWM	CTCM	Timer/Counter Mode Setting.

Table 893:

Attribute	Type	Allowed Values	Default	Description
TC_SCLK_SEL	String	PCLOCK, POSC, NCLOCK, NOSC	PCLOCK	Timer/Counter Input Clock Setting.
TC_CCLK_SEL	Integer	0, 1, 8, 64, 256, 1024	1	Timer/Counter Input Clock Divider Setting.
TC_TOP_SET	Integer	0 to 65535	65535	Specifies the Timer/Counter Top Set value.
TC_OCR_SET	Integer	0 to 65535	32767	Specified the Timer/Counter OCR Set value.
TC_OC_MODE	String	STATIC, TOGGLE, WAVE_GENERATOR, INV_WAVE_GENERATOR	TOGGLE	Mode Selection on TC_OC waveform generation.
TC_RESETN	Boolean	ENABLED, DISABLED	ENABLED	Timer/Counter reset enable.
TC_TOP_SEL	Boolean	ON, OFF	ON	Sets Top Counter value.
TC_OV_INT	Boolean	ON, OFF	OFF	Timer/Counter overflow interrupt request.
TC_OCR_INT	Boolean	ON, OFF	OFF	Timer/Counter interrupt request.
TC_ICR_INT	Boolean	ON, OFF	OFF	The input capture interrupt.
TC_OVERFLOW	Boolean	ENABLED, DISABLED	ENABLED	Enables Timer/Counter for an overflow tag.
TC_ICAPTURE	Boolean	ENABLED, DISABLED	DISABLED	Enables Timer/Counter for input capture.
TIMEOUT	String	BYPASS, USER, COUNTER	BYPASS	Specifies the stop to standby delay.
UDS_TRN	128-bit value	Any 128-bit value	0	Unique Device Secret. Setting to all 0s disables the UDS feature.
UDS_TRN_FORMAT		BIN, HEX, ASCII	ASCII	Format for Unique Device Secret.
UFMx_INIT_PAGES	Integer	0 to Max Number of Pages in UFM array.	0	Specifies the number of pages with initialization data for UFMx, where x holds value between 0 to 3. If the user doesn’t enter a value, it is set to 0.
UFMx_INIT_START_PAGE	Integer	Calculated by software.	0	Software calculates and displays the starting page of the init data in the UFM array, where x holds value between 0 to 3. Read only field.
UFMx_INIT_ALL_ZEROS	Boolean	ENABLED, DISABLED	ENABLED	Specifies that the User Flash Memory is initialized with all 0s. x can hold a value between 0 to 3. By default, the radio button is selected.

Table 893:

Attribute	Type	Allowed Values	Default	Description
UFMx_INIT_FILE_NAME	String	String	None	Uploads the User Flash Memory Initialization data file, where x holds value between 0 to 3. By default, the radio button is not selected. The browse button is not enabled if the radio button is not selected.
UFMx_INIT_FILE_FORMAT	String	HEX, BIN	HEX	Specifies the selection of file format, either Binary or Hexadecimal, x holds a value between 0 and 3.
UPDT	String	POS, NEG	POS	Attached to a DDR-type primitive (such as ODDRX4A), the UPDT attribute takes the POS and NEG options, which allows you to update block output.
WAIT_FOR_EDGE	Boolean	ENABLED, DISABLED	ENABLED	Wait for edge.
WAKE_ON_LOCK	Boolean	ON, OFF	ON, OFF	This is a legacy attribute and not supported for new configurations. Attached to a PLL-type primitive.
WAKEUP	String	USER, CFG, USER_CFG	USER_CFG	Specifies the wake option for wake signals. There are three options. ▶ USER: In this case, MAP checks for the connection to the USERSTDBY pin only. If the pin is not driven by a signal that can be toggled, MAP issues error with DRC for this case only. ▶ CFG: In this case, MAP checks for JTAG, I2C, and SLAVE_SPI. If all are disabled, MAP errors out with config mode error. ▶ USER_CFG: In this case, MAP checks the USERSTDBY pin connection and JTAG, I2C, and SLAVE_SPI. Map errors out only when the USERSTDBY pin is not driven by live signal and all settings are disabled.

Table 893:

Attribute	Type	Allowed Values	Default	Description
WRITEMODE	String	NORMAL, WRITETHROUGH, READBEFORE	NORMAL	Specifies Read/Write mode. Attached to a dual- and single-port RAM primitive. WRITEMODE_A and WRITEMODE_B are used for dual-port RAM primitives and refer to the A and B ports.
WRITE_LEVELING	String	0T, 1T, 2T	2T	Sets the Write Leveling. 0T: for DDR2, or DDR3 without WL. 1T: For DDR3 with 1T. 2T: for DDR3 with 2T range.

Chapter 20

Command Line Reference Guide

This help guide contains information necessary for running the core design flow development from the command line. For tools that appear in the Diamond graphical user interface, use Tcl commands to perform commands that are described in the “Tcl Command Reference Guide” on page 2586.

Command Line Program Overview

Lattice FPGA command line programs can be referred to as the FPGA flow core tools. These are tools necessary to run a complete design flow and are used for tasks such as module generation, design implementation, design verification, and design configuration. This topic provides an overview of those tools, their functions, and provides links to detailed usage information on each tool.

Each command line program provides multiple options for specifying device information, applying special functions using switches, designating desired output file names, and using command files. The programs also have particular default behavior often precludes the need for some syntax, making commands less complex. See “Command Line General Guidelines” on page 2467 and “Command Line Syntax Conventions” on page 2469.

To learn more about the applications, usage, and syntax for each command line program, click on the hyperlink of the command line name in the section below.

Note

Many of the command line programs described in this topic are run in the background when using the tools you run in the Diamond environment. Please also note that in some cases, command line tools described here are used for earlier FPGA architectures only, are not always recommended for command line use, or are only available from the command line.

Design Implementation Using Command Line Tools The table below shows all of the command line tools used for various design functions, their

graphical user interface counterparts, and provides functional descriptions of each.

Table 894: Diamond Core Design and Tool Chart

Design Function	Command Line Tool	Diamond Process	Description
Core Implementation and Auxiliary Tools			
Simulation	cmpl_libs.tcl	Simulation	Simulation libraries compilation.
Synthesis	SYNTHESIS	Synthesis Design	(*MachXO/XO2 Only*) Runs input source files though synthesis based on Lattice Synthesis Engine options set in Strategies.
Synthesis	synpwrap	Synthesis Design	Used to manage Synplicity Synplify and Synplify Pro synthesis programs.
Netlist Translation and Conversion	EDIF2NGD NGDBUILD	Translate Database	Converts netlists into generic databases which use native primitive terms that the system can interpret and implement.
Mapping	MAP	Map Design	Converts a design represented in logical terms into a network of physical components or configurable logic blocks.
Placement and Routing	PAR	Place & Route Design	Assigns physical locations to mapped components and creates physical connections to join components in an electrical network.
Static Timing Analysis	TRCE	MAP TRACE Report, Place & Route TRACE Report	Generates reports that can be used for static timing analysis. Also known as TRACE.
I/O Timing Analysis	IOTIMING	I/O Timing Analysis	Generates a report that provides port information regarding setup/hold time requirements and min/max clock-to-out delay. Also determines the clocks and their associated data ports.
I/O SSO (Simultaneous Switching Output) Analysis	SSOANA	SSO Analysis	Generates an SSO analysis report (.sso) based on the SSO preferences you specified in your LPF file.

Table 894: Diamond Core Design and Tool Chart

Design Function	Command Line Tool	Diamond Process	Description
Backannotation	LDBANNO	Tool does not exist in Diamond interface as process but employed in file export.	Distributes the physical design information back to the logical design to generate a timing simulation file.
Convert Logical PCS Attributes to Physical PCS Attributes (LatticeEPC3 only)	LTXT2PTXT	Not applicable.	Converts Logical PCS Attributes to Physical PCS Attributes (LatticeEPC3 only)
Bitstream Generation	BITGEN	Bitstream	Converts a fully routed physical design into configuration bitstream data.
Device Programming	PGRCMD	Device Programming	Downloads data files to an FPGA device.
Deployment Tool	DDTCMD	Deployment Tool	Converts data files to other formats. Also, uses the data files produced to generate other data file formats.

See Also
- "Command Line Data Flow" on page 2467
- "Command Line General Guidelines" on page 2467
- "Command Line Syntax Conventions" on page 2469
- "Invoking Core Tool Command Line Tool Help" on page 2471

Command Line Basics

This section provides basic instructions for running any of the core design flow development and tools from the command line.

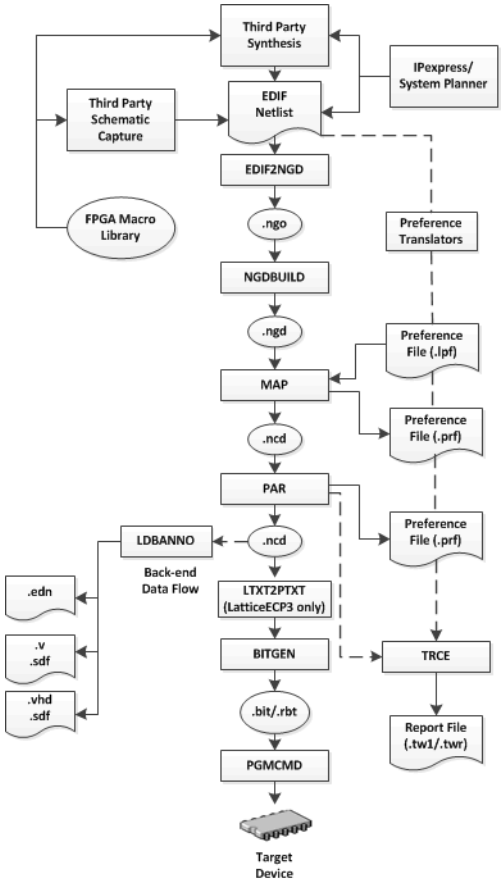
Topics include:

- "Command Line Data Flow" on page 2467
- "Command Line General Guidelines" on page 2467
- "Command Line Syntax Conventions" on page 2469
- "Setting Up the Environment to Run Command Line" on page 2470
- "Invoking Core Tool Command Line Programs" on page 2471
- "Invoking Core Tool Command Line Tool Help" on page 2471
- "Reading NCD Files" on page 2472
- "Sample Text-Converted NCD File" on page 2473

Command Line Data Flow

The following chart illustrates the FPGA command line tool data flow through a typical design cycle.

Command Line Tool Data Flow



See Also ▶"Command Line Reference Guide" on page 2464

▶ "Command Line General Guidelines" on page 2467

Command Line General Guidelines

You can run the FPGA family Diamond design tools from the command line. The following are general guidelines that apply.

- Files are position-dependent. Generally, they follow the convention [options] <infile> <outfile> <preference_file> (although order of <outfile> and <infile> are sometimes reversed). Use the **-h** command line option to check the exact syntax; for example, **par -h**.

- For any Diamond FPGA command line program, you can invoke help on available options with the **-h** or **-help** command. See "Invoking Core Tool Command Line Programs" on page 2471 for more information.
- Command line options are entered on the command line in any order, preceded by a hyphen (-), and separated by spaces.
- Most command line options are case-sensitive and must be entered in lowercase letters. When an option requires an additional parameter, the option and the parameter must be separated by spaces or tabs (i.e., **-l 5** is correct, **-l5** is not).
- Options that do not require an additional parameter may be grouped together preceded by a single hyphen (i.e., **-arw** is the same as **-a -r -w**).
- Options can appear anywhere on the command line. Arguments that are bound to a particular option must appear after the option (i.e., **-f** <command_file> is legal; <command_file> **-f** is not).
- For options that may be specified multiple times, in most cases the option letter must precede each parameter. For example, **-b romeo juliet** is not acceptable, while **-b romeo -b juliet** is acceptable.
- If you enter the FPGA family Diamond application name on the command line with no arguments and the application requires one or more arguments (**par,** for example), you get a brief usage message consisting of the command line format string.
- For any Diamond FPGA command line program, you can store program commands in a command file. Execute an entire batch of arguments by entering the program name, followed by the **–f** option, and the command file name. This is useful if you frequently execute the same arguments each time you execute a program or to avoid typing lengthy command line arguments. See "Using Command Files" on page 2582.

See Also

- "Command Line Reference Guide" on page 2464
- "Invoking Core Tool Command Line Tool Help" on page 2471
- "Command Line Syntax Conventions" on page 2469
- "Using Command Files" on page 2582
- "Command Line Data Flow" on page 2467

Command Line Syntax Conventions

The following conventions are used when commands are described:

Table 895: Command Line Syntax Conventions

Convention	Meaning
( )	Encloses a logical grouping for a choice between sub-formats.
[ ]	Encloses items that are optional. (Do not type the brackets.) Note that <infile[**.ncd**]> indicates that the .ncd extension is optional but that the extension must be NCD.
{ }	Encloses items that may be repeated zero or more times.
|	Logical OR function. You must choose one or a number of options. For example, if the command syntax says **pan up | down | right | left** you enter **pan up** or **pan down** or **pan right** or **pan left**.
< >	Encloses a variable name or number for which you must substitute information.
, (comma)	Indicates a range for an integer variable.
- (dash)	Indicates the start of an option name.
:	The bind operator. Binds a variable name to a range.
bold text	Indicates text to be taken literally. You type this text exactly as shown (for example, “Type **autoroute -all -i 5** in the command area.”) Bold text is also used to indicate the name of an EPIC command, a UNIX command, or a DOS command (for example, “The **playback** command is used to execute the macro you created.”).
Italic text or text enclosed in angle brackets <>	Indicates text that is not to be taken literally. You must substitute information for the enclosed text. Italic text is also used to show a file directory path, for example, “the file is in the /cd/data/diamond directory”).
Monospace	Indicates text that appears on the screen (for example, “File already exists.”) and text from UNIX or DOS text files. Monospace text is also used for the names of documents, files, and file extensions (for example, “Edit the autoexec.bat file” or “Native Circuit Description (.ncd) files.”

See Also

- “Command Line Reference Guide” on page 2464
- “Command Line General Guidelines” on page 2467
- “Invoking Core Tool Command Line Tool Help” on page 2471
- “Using Command Files” on page 2582

Setting Up the Environment to Run Command Line

For Windows The environments for both the Diamond Tcl Console window or Diamond Standalone Tcl Console window (pnmainc.exe) are already set. You can start entering Tcl tool command or core tool commands in the console and the software will perform them.

When running Diamond from the Windows command line (via cmd.exe), you will need to add the appropriate values to the following environment variables:

- PATH (for 32-bit systems)

  `<Install_directory>\bin\nt;<Install_directory>\ispfpga\bin\nt;`

  Example:

  `c:\lscc\diamond\3.10\bin\nt;c:\lscc\diamond\3.10\ispfpga\bin\nt;`

- PATH (for 64-bit systems)

  `<Install_directory>\bin\nt64;<Install_directory>\ispfpga\bin\nt64;`

  Example:

  `c:\lscc\diamond\3.10\bin\nt64;c:\lscc\diamond\3.10\ispfpga\bin\nt64;`

- FOUNDRY (all systems)

  `<Install_directory>\ispfpg`

  Example:

  `set FOUNDRY = c:\lscc\diamond\3.10_x64\ispfpga`

For Linux On Linux, Diamond software provides a similar standalone Tcl Console window (pnmainc) that sets the environment. The user can enter Tcl commands and core tool commands in it.

If you do not use the Tcl Console window, before running any of the command line programs, you need to run the following command:

- For CSH (64-bit systems):

  ```
  cd <install_path>/diamond/3.10/bin/lin64
  source setupenv
  ```

- For BASH (64-bit):

  ```
  export bindir=<Install_directory>/bin/lin64
  source $bindir/diamond_env
  ```

After setting up for either Windows or PC, you can run the Diamond executable files directly. For example, you can invoke the Place and Route program by:

```
par -w test_map.ncd test.ncd test.prf
```

See Also ▶"Invoking Core Tool Command Line Programs" on page 2471

- "Invoking Core Tool Command Line Tool Help" on page 2471

Invoking Core Tool Command Line Programs

This topic provides general guidance for running the Diamond FPGA flow core tools. Refer to "Command Line Program Overview" on page 2464 to see what these tools include and for further information.

For any Diamond FPGA command line programs, you begin by entering the name of the command line program followed by valid options for the program separated by spaces. Options include switches (**-f**, **-p**, **-o**, etc.), values for those switches, and file names, which are either input or output files. You start command line programs by entering a command in the UNIX™ or DOS™ command line. You can also run command line scripts or command files.

See the table in the section Design Implementation Using Command Line Tools in "Command Line Reference Guide" on page 2464 for details and links to specific information on usage and syntax. You will find all of the usage information on the command line in the **Running FPGA Tools from the Command Line > Command Line Tool Usage** book topics.

See Also ▶"Command Line Reference Guide" on page 2464

- "Command Line Syntax Conventions" on page 2469
- "Invoking Core Tool Command Line Tool Help" on page 2471
- "Setting Up the Environment to Run Command Line" on page 2470
- "Using Command Files" on page 2582

Invoking Core Tool Command Line Tool Help

To get a brief usage message plus a verbose message that explains each of the options and arguments, enter the FPGA family Diamond application name on the command line followed by **-help** or **-h**. For example, enter **bitgen -h** for option descriptions for the **bitgen** program.

To redirect this message to a file (to read later or to print out), enter this command:

command_name -help | -h > filename

The usage message is redirected to the filename that you specify.

For those FPGA family Diamond applications that have architecture-specific command lines (e.g., or2c00a), you must enter the application name, **-help** (or **-h**), and the architecture to get the verbose usage message specific to that architecture. If you fail to specify the architecture, you get a message similar to the following:

Use '<appname> -help <architecture>' to get detailed usage for a particular architecture.

See Also ▶"Command Line Reference Guide" on page 2464

- "Command Line Data Flow" on page 2467
- "Command Line General Guidelines" on page 2467
- "Command Line Syntax Conventions" on page 2469
- "Setting Up the Environment to Run Command Line" on page 2470
- "Using Command Files" on page 2582

Reading NCD Files

If you need to read a Native Circuit Description (.ncd) file, that is a physical design NCD file, you can use the NCDREAD utility program that is available in the <install_dir>\ispFPGA\bin\nt | sol file path for PC and UNIX/Linux platforms respectively. This utility is only available from the command line.

NCD files represent your design as a network of device-specific components (e.g., PFUs, PFFs, and EBRs) or configurable logic blocks in the form of a physical design (NCD). These files are the output of the **Map Design** (**map**) and **Place & Route Design** (**par**) processes in Diamond.

Using NCDREAD, you can quickly generate an ASCII (text) file based on the data found in one or more NCD files.

To run NCDREAD from the UNIX or DOS command line:

- Type in the **ncdread** command with valid options and enter. Syntax for this utility is as follows:

ncdread [**–o** <outfile_name>] <infile1[.**ncd**]> {<infile2[.**prf**]> ...}

where the .prf file is the project preference file.

Note

If you do not specify the **–o** option to write the output to a file, standard output goes to your screen.

Example 1 An example of an actual command line argument might be as follows:

ncdread –o readfile.txt arcrad4.ncd ramus256.prf

Example 2 You may want to find an HDL instance name of a given memory, you could use the NCDREAD program utility on your NCD file to generate a text file containing those names. Note however that the Memory Generation Tool in Diamond automates this process. For example,

ncdread rom8x8.ncd -o rom8x8.out

In the above command line, an NCD file named **rom8x8** is used as input for NCDREAD, outputting a text file called **rom8x8.out**.

Now, look in the **rom8x8.out** text file for the instance name using a text editing tool to search on the word “INITVAL” in the Config String. When you find an occurrence of INITVAL, look for the corresponding COMP name in the line right above it as shown in the example below:

```
NC_COMP:20 - <TEST/rom8x8_0_0_0>, site = EBR_R10C6
```

In the above COMP name, **TEST** is the instance name. The **rom8x8_0_0_0** value is generated during module generation and is based upon how many EBRs this particular memory takes. So, if the memory takes two EBRs, then there will be two NC_COMPs with the name values of **TEST/rom8x8_0_0_0** and **TEST/rom8x8_0_0_1**. In this case, the instance name would be still be **TEST**. See “Sample Text-Converted NCD File” on page 2473if you wish to test this search on INITVAL.

See Also ▶“Sample Text-Converted NCD File” on page 2473

- “Creating a Memory Initialization File with Memory Generator” on page 203
- “Memory Initialization File Format” on page 204

Sample Text-Converted NCD File

This topic shows a sample of what a Native Circuit Description (.ncd) file looks like after being converted to text using the NCDREAD utility.

Sample Text-Converted NCD File

```
NC_DESIGN <top> - version 3.0
  vendor = LATTICE, package = FPBGA672, peformance = 3
  72 Properties -
    IOBUSIDX:0 long 7
    IOBUSIDX:1 long 6
    IOBUSIDX:10 long 5
    IOBUSIDX:11 long 4
    IOBUSIDX:12 long 3
    IOBUSIDX:13 long 2
    IOBUSIDX:14 long 1
    IOBUSIDX:15 long 0
    IOBUSIDX:2 long 5
    IOBUSIDX:3 long 4
    IOBUSIDX:4 long 3
    IOBUSIDX:5 long 2
```

```
    IOBUSIDX:6 long 1
    IOBUSIDX:7 long 0
    IOBUSIDX:8 long 7
    IOBUSIDX:9 long 6
    IOBUSNAME:0 string
      "Address(7:0)"
    IOBUSNAME:1 string
      "Address(7:0)"
    IOBUSNAME:10 string
      "Q(7:0)"
    IOBUSNAME:11 string
      "Q(7:0)"
   *** Properties listing truncated here for example purposes
***

  21 comps
    NC_COMP:0 - <TEST/SLICE_0>, site = R11C6B
            Config String: <REGMODE:#OFF MODE:LOGIC K1:#OFF
K0::H0=1 REG1:#OFF
            REG0:#OFF Q1:#OFF Q0:#OFF F1:#OFF F0:F GSR:#OFF
M0MUX:#OFF
            CLKMUX:#OFF CEMUX:#OFF LSRMUX:#OFF OFX1:#OFF
SRMODE:#OFF OFX0:#OFF
            LSRONMUX:#OFF M1MUX:#OFF ALU2_MULT:#OFF FCO:#OFF
FCIMUX:#OFF
            DI1MUX:#OFF DI0MUX:#OFF CCU2:#OFF>

        25 pins -
          pin 19 - F0: <TEST/VCC>
        Logical names -
          Primitive K0 = <TEST/VCC_0>
        1 Properties -
          NGID0 long 10
    NC_COMP:1 - <Q_0>, site = G7
            Config String: <PADDI:#OFF TRIMUX:#OFF
IOBUF:::DRIVE=12 DATAMUX:OP0
            ODMUX:1:::1=0 VREF:OFF>
        5 pins -
          pin 4 - PADDO: <Q_c_0>
        Logical names -
          Primitive IOBUF = <Q_pad_0>
          Primitive PAD = <Q_0>
        1 Properties -
          #%PAD%PINID long 15
    NC_COMP:2 - <Address_0>, site = F6
            Config String: <PADDI:PADDI TRIMUX:#OFF IOBUF:#ON
DATAMUX:#OFF
            ODMUX:#OFF VREF:OFF>
        5 pins -
          pin 0 - PADDI: <Address_c_0>
        Logical names -
          Primitive IOBUF = <Address_pad_0>
          Primitive PAD = <Address_0>
        1 Properties -
          #%PAD%PINID long 7
    NC_COMP:3 - <Q_7>, site = D8
            Config String: <PADDI:#OFF TRIMUX:#OFF
IOBUF:::DRIVE=12 DATAMUX:OP0
            ODMUX:1:::1=0 VREF:OFF>
```

```
        5 pins -
          pin 4 - PADDO: <Q_c_7>
        Logical names -
          Primitive IOBUF = <Q_pad_7>
          Primitive PAD = <Q_7>
        1 Properties -
          #%PAD%PINID long 8
  *** COMP name listing truncated here for example purposes ***
    NC_COMP:20 - <TEST/rom8x8_0_0_0>, site = EBR_R10C6
           Config String: <MODE:DP8KA SP8KA:#OFF PDP8KA:#OFF

DP8KA:::DATA_WIDTH_A=18,DATA_WIDTH_B=18,REGMODE_A=OUTREG,REGMOD
E_B=OUTREG,

INITVAL_00=0x000F0000DE000BC0009A000780005600034000012000F0000DE
000BC0009A00078000560003400012,

INITVAL_01=0x00000000000000000000000000000000000000000000F0000DE
000BC0009A00078000560003400012:

WEA=0,CSA0=0,CSA1=0,CSA2=0,WEB=0,CSB0=0,CSB1=0,CSB2=0,DIA0=0,DI
A1=0,DIA2=0,DIA3=0,DIA4=0,DIA5=0,

DIA6=0,DIA7=0,DIA8=0,DIA9=0,DIA10=0,DIA11=0,DIA12=0,DIA13=0,DIA
14=0,DIA15=0,DIA16=0,DIA17=0,ADA0=0,

ADA1=0,ADA12=0,DIB0=0,DIB1=0,DIB2=0,DIB3=0,DIB4=0,DIB5=0,DIB6=0
,DIB7=0,DIB8=0,DIB9=0,DIB10=0,DIB11=0,

DIB12=0,DIB13=0,DIB14=0,DIB15=0,DIB16=0,DIB17=0,ADB2=0,ADB3=0>
        114 pins -
          pin 37 - ADA11: <Address_c_7>
          pin 38 - ADA10: <Address_c_6>
          pin 39 - ADA9: <Address_c_5>
          pin 40 - ADA8: <Address_c_4>
          pin 41 - ADA7: <Address_c_3>
          pin 42 - ADA6: <Address_c_2>
          pin 43 - ADA5: <Address_c_1>
          pin 44 - ADA4: <Address_c_0>
          pin 45 - ADA3: <TEST/VCC>
          pin 46 - ADA2: <TEST/VCC>
          pin 49 - ADB12: <TEST/VCC>
          pin 50 - ADB11: <Address_c_7>
          pin 51 - ADB10: <Address_c_6>
          pin 52 - ADB9: <Address_c_5>
          pin 53 - ADB8: <Address_c_4>
          pin 54 - ADB7: <Address_c_3>
          pin 55 - ADB6: <Address_c_2>
          pin 56 - ADB5: <Address_c_1>
          pin 57 - ADB4: <Address_c_0>
          pin 60 - ADB1: <TEST/VCC>
          pin 61 - ADB0: <TEST/VCC>
          pin 62 - CEA: <OutClockEn_c>
          pin 67 - RSTA: <Reset_c>
          pin 68 - CEB: <OutClockEn_c>
          pin 73 - RSTB: <Reset_c>
          pin 74 - CLKA: <OutClock_c>
          pin 75 - CLKB: <OutClock_c>
          pin 76 - MORCLKA: <OutClock_c>
```

```
      pin 77 - MORCLKB: <OutClock_c>
      pin 96 - DOA0: <Q_c_0>
      pin 97 - DOA1: <Q_c_1>
      pin 98 - DOA2: <Q_c_2>
      pin 99 - DOA3: <Q_c_3>
      pin 100 - DOA4: <Q_c_4>
      pin 101 - DOA5: <Q_c_5>
      pin 102 - DOA6: <Q_c_6>
      pin 103 - DOA7: <Q_c_7>
    260 Properties -
      #%DP8KA%PINNAME:0 string
        "DOA0"
      #%DP8KA%PINNAME:1 string
        "DOA1"
      #%DP8KA%PINNAME:10 string
        "DOA10"
      #%DP8KA%PINNAME:100 string
        "RSTA"
      #%DP8KA%PINNAME:101 string
        "RSTB"
      #%DP8KA%PINNAME:102 string
        "CSA0"
      #%DP8KA%PINNAME:103 string
        "CSA1"
      #%DP8KA%PINNAME:104 string
        "CSA2"
      #%DP8KA%PINNAME:105 string
        "CSB0"
      #%DP8KA%PINNAME:106 string
        "CSB1"
      #%DP8KA%PINNAME:107 string
        "CSB2"
      #%DP8KA%PINNAME:108 string
        "CLKA"
      #%DP8KA%PINNAME:109 string
        "CLKB"
      #%DP8KA%PINNAME:11 string
        "DOA11"
      #%DP8KA%PINNAME:110 string
        "CEA"
      #%DP8KA%PINNAME:111 string
        "CEB"
      #%DP8KA%PINNAME:12 string
        "DOA12"
      #%DP8KA%PINNAME:13 string
        "DOA13"
*** COMP name listing truncated here for example purposes ***

0 macdefs
0 macinsts
0 black boxes
0 hardwire signals
```

Command Line Tool Usage

This section contains usage information of all of the command line tools and valid syntax descriptions for each.

Topics include:

- "Running cmpl_lib.tcl from the Command Line" on page 2477
- "Running SYNTHESIS from the Command Line" on page 2480
- "Running EDIF2NGD from the Command Line" on page 2487
- "Running NGDBUILD from the Command Line" on page 2492
- "Running MAP from the Command Line" on page 2495
- "Running PAR from the Command Line" on page 2505
- "Running MPARTRCE from the Command Line" on page 2522
- "Running TRACE from the Command Line" on page 2526
- "Running I/O Timing from the Command Line" on page 2531
- "Running Backannotation from the Command Line" on page 2533
- "Running LTXT2PTXT from the Command Line (ECP5 and LatticeECP3 Only)" on page 2537
- "Running Bit Generation from the Command Line" on page 2538
- "Running Programmer from the Command Line" on page 2549
- "Running the Deployment Tool from the Command Line" on page 2552
- "Running I/O SSO Analysis from the Command Line" on page 2578
- "Running Various Utilities from the Command Line" on page 2579
- "Using Command Files" on page 2582
- "Using Command Line Shell Scripts" on page 2584

Running cmpl_lib.tcl from the Command Line

The cmpl_lib.tcl command allows you to perform simulation library compliation from the command line.

The following information is for running cmpl_libs.tcl from the command line using the tclsh application. The supported TCL version is 8.5 or higher.

If you don't have TCL installed, or you have an older version, perform the following:

- Add <*Diamond_install_path*>/tcltk/bin to the front of your PATH, and

- For Linux users only, add <*Diamond_install_path*>/tcltk/lib to the front of your LD_LIBRARY_PATH

Note

The default version of TCL on Linux could be older and may cause the script to fail. Ensure that you have TCL version 8.5 or higher.

To check TCL version, type:

```
tclsh
% info tclversion
% exit
```

For script usage, type:

```
tclsh cmpl_libs.tcl [-h|-help|]
```

Note

If Modelsim/Questa is already in your PATH and preceding any Aldec tools, you can use:

'-sim_path .' for simplification; '.' will be added to the front of your PATH.

Note

Ensure the FOUNDRY environment variable is set. If the FOUNDRY environment variable is missing, then you need to set it before running the script. For details, refer to "Setting Up the Environment to Run Command Line" on page 2470.

Check log files under <*target_path*> (defualt = .) for any errors, as follows:

- For Linux, type:

  ```
  grep -i error *.log
  ```

- For Windows, type:

  ```
  find /i "error" *.log
  ```

Subjects included in this topic:

- Running compl_lib.tcl
- Command Line Syntax
- cmpl_lib.tcl Options
- Examples

Running compl_lib.tcl cmpl_libs.tcl allows you to compile simulation libraries from the command line.

Command Line Syntax tclsh <*diamond_install_path*>/cae_library/simulation/script/cmpl_libs.tcl -sim_path <sim_path> [-sim_vendor {mentor<default>}] [-lang {verilog|vhdl|all<default>}] [-device {sc|scm|ec|xp|ecp|machxo|ecp2|ecp2m|xp2|ecp3|machxo2|machxo3d|machxo3l|lptm|lptm2|ecp5u|ecp5um|lifmd|lifmdf|all<default>}] [-target_path <target_path>]

cmpl_lib.tcl Options The table below contains all valid options for cmpl_libs.tcl

Table 896: cmpl_lib.tcl Command Line Options

Option	Description
-sim_path <sim_path> [-sim_vendor {mentor<default>}] [-lang {verilog|vhdl|all<default>}] [-device {sc|scm|ec|xp|ecp|machxo|ecp2|ecp2m|xp2|ecp3|machxo2|machxo3d|machxo3l|lptm|lptm2|ecp5u|ecp5um|lifmd|lifmdf|all<default>}] [-target_path <target_path>]	▶ The -sim_path argument specifies the path to the simulation tool executable (binary) folder. This option is mandatory. Currently only Modelsim and Questa simulators are supported. NOTE: If <*sim_path*> has spaces, then it must be surrounded by " ". Do not use { }. ▶ The -sim_vendor argument is optional, and intended for future use. It currently supports only Mentor Graphics simulators (Modelsim / Questa). ▶ The -lang argument specifies the HDL type of the compiled libraries. This argument is optional, and the default is to compile both types of libraries. ▶ The -device argument specifies the Lattice FPGA device to compile simulation libraries for. This argument is optional, and the default is to compile libraries for all the Lattice FPGA devices. ▶ The -target_path argument specifies the target path, where you want the compiled libraries and modelsim.ini file to be located. This argument is optional, and the default target path is the current folder. NOTES: (1) This argument is recommended if the current folder is the Diamond's startup (binary) folder, or if the current folder is write-protected. (2) If <*target_path*> has spaces, then it must be surrounded by " ". Do not use { }.

Examples This section illustrates and describes a few examples of Simulation Libraries Compilation Tcl command.

Example 1 The following command will compile all the Lattice FPGA libraries for both Verilog and VHDL simulation, and place them under the folder specified by -target_path. The path to Modelsim is specified by -sim_path.

```
tclsh <Diamond_install_path>/cae_library/simulation/script/
cmpl_libs.tcl -sim_path C:/modeltech64_10.0c/win64 -target_path
c:/mti_libs
```

Example 2 The following command will compile the LatticeECP3 libraries for Verilog simulation only, and place them under the current folder. The path to Modelsim is specified by -sim_path.

```
tclsh <Diamond_install_path>/cae_library/simulation/script/
cmpl_libs.tcl -sim_path C:\modeltech_6.6d\win32 -device ecp3 -
lang verilog
```

Note

Backslash (\) is for Windows paths only. For Linux, use forward slash (/).

Running SYNTHESIS from the Command Line

The Lattice synthesis tool SYNTHESIS allows you to synthesize source schematics, Verilog and VHDL HDL source files into netlists for design entry into the Diamond environment. Based on your strategy settings you specify in Diamond, a synthesis project (.synproj) file is created and then used by SYNTHESIS using the -f option. Diamond translates strategy options into command line options described in this topic.

Verilog source files are passed to the program using the **-ver** option and VHDL source files are passed using the **-vhd** option. For mixed language designs the language type is automatically determined by SYNTHESIS based on the top module of the design. For IP design, you must also specify IP location (**-ip_dir)**, IP core name (**-corename**), and encrypted RTL file name (**-ertl_file**).

For synthesis output, NGD and NGO formats cannot be used together in the same command. You must specify one or the other format. If NGO output is selected, the NGD file must be generated separately using the NGDBUILD command together with any other NGO files that may be required (e.g., PMIs).

Subjects included in this topic:

- Running SYNTHESIS
- Command Line Syntax
- SYNTHESIS Options
- Examples

Running SYNTHESIS SYNTHESIS will convert your input netlist (.ed*, .v, .ngo) file into an .ngo file that is used for input for NGDBUILD.

- To run SYNTHESIS, type **synthesis** on the command line with valid options. A sample of a typical SYNTHESIS command would be as follows:

  synthesis -ngd v_top.ngd

There are many command line options that give you control over the way SYNTHESIS processes the output file. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **synthesis** [**-a** <arch>] [**-d** <device>] [**-f** <file_name>][**-t** <package>] [**-p** <searchpath>] [**-top** <module_name>] [**-lib** <VHDL lib name>] [**-vhd** <VHDL files>] [**-ver** <Verilog files>] [**-hdl_param** <name,value>] [**-ngd** <file_name.ngd>] [**-ngo** <file_name.ngo>] [**-optimization_goal** <balanced | area | timing>] [**-force_gsr** <auto | yes | no>] [**-ramstyle** <auto | distributed | block_ram | registers>] [**-romstyle** <auto | logic | EBR>] [**-output_edif** <file_name.edf>] [**-sdc** <sdc_file.ldc>] [**-lpf** <pref_file.lpf>] [**-loop_limit** <value>] [**-logfile** <file_name>] [**-frequency** <value>] [**-max_fanout** <value>] [**-fsm_encoding_style** <binary | one-hot | gray>][**-bram_utilization** <value>] [**-mux_style** <auto | pfu_mux | L6Mux_single | L6Mux_multiple>] [**-use_carry_chain** <0|1>] [**-carry_chain_length** <chain_length>] [**-use_io_insertion** <0|1>] [**-use_io_reg** <0|1>] [**-resource_sharing** <0|1>] [**-propagate_constants** <0|1>] [**-remove_duplicate_regs** <0|1>] [**-ip_dir** <dir_name>] [**-corename** <core_name>] [**-ertl_file** <file_name>] [**-twr_paths** <num_paths >] [**-ifd**] [**-dt**] [**-fix_gated_clocks** <fix_gated_clocks_value>] [**-vh2008**] [**-key** <key_file>]

SYNTHESIS Options The table below contains descriptions of all valid options for SYNTHESIS.

Table 897: SYNTHESIS Command Line Options

Option	Description
-a <arch>	Sets the FPGA architecture. This synthesis option must be specified and if the value is set to any unsupported FPGA device architecture the command will fail.
-d <device>	Specifies the device type for the architecture (optional).
-f <file_name>	Specifies the synthesis project file name (.synproj). The project file can be edited by the user to contain all desired command line options.
-s <device>	Specifies the device peformance grade for the architecture (optional).
-t <package>	Specifies the package type of the device.
-p <searchpath>	Add searchpath for NGO files, Verilog “include” files (optional).
-top <module_name>	Name of top module (optional, but better to have to avoid ambiguity).
-lib <VHDL lib name>	Name of VHDL library (optional).
-vhd <VHDL files>	Names of VHDL design files (must have, if language is VHDL or mixed language).
-ver <Verilog files>	Names of Verilog design files (must have, if language is Verilog, or mixed language).
-hdl_param <name, value>	Allows you to override HDL parameter pairs in the design file.

Table 897: SYNTHESIS Command Line Options

Option	Description
-ngd <ngd_file>	Specifies the name of the NGD file output.
-ngo <ngo_file>	Specifies the name of the NGO file output. This is an optional switch that cannot be used in conjunction with the **-ngd** option.
-optimization_goal <balanced | area | timing>	The synthesis tool allows you to choose among the following optimization options: ▶ **balanced** balances the levels of logic. ▶ **area** optimizes the design for area by reducing the total amount of logic used for design implementation. ▶ **timing** optimizes the design for timing. The default setting depends on the device type. Smaller devices, such as MachXO and Platform Manager, default to balanced. Larger devices—ECP5U, ECP5UM, LatticeECP2, LatticeECP3, and LatticeXP2—default to timing.
-force_gsr <auto | yes | no>	Enables (**yes**) or disables (**no**) forced use of the global set/reset routing resources. When the value is auto, the synthesis tool decides whether to use the global set/reset resources. The default behavior is determined by the switch settings: When **-ngo** or **-output_edif** then the default is no. When **-ngd** then default is auto.

Table 897: SYNTHESIS Command Line Options

Option	Description
-ramstyle <auto | distributed | block_ram | registers>	Sets the type of random access memory globally to *distributed*, *embedded block RAM*, or *registers*.The default is auto which attempts to determine the best implementation, that is, synthesis tool will map to technology RAM resources (EBR/Distributed) based on the resource availability. This option will apply a syn_ramstyle attribute globally in the source to a module or to a RAM instance. To turn off RAM inference, set its value to registers. ▶ **registers** causes an inferred RAM to be mapped to registers (flip-flops and logic) rather than the technology-specific RAM resources. ▶ **distributed** causes the RAM to be implemented using the distributed RAM or PFU resources. ▶ **block_ram** causes the RAM to be implemented using the dedicated RAM resources. If your RAM resources are limited, for whatever reason, you can map additional RAMs to registers instead of the dedicated or distributed RAM resources using this attribute. ▶ **no_rw_check** (Certain technologies only). You cannot specify this value alone. Without no_rw_check, the synthesis tool inserts bypass logic around the RAM to prevent the mismatch. If you know your design does not read and write to the same address simultaneously, use no_rw_check to eliminate bypass logic. Use this value only when you cannot simultaneously read and write to the same RAM location and you want to minimize overhead logic.

Table 897: SYNTHESIS Command Line Options

Option	Description
-romstyle <auto | logic | EBR>	Allows you to globally implement ROM architectures using *dedicated* , *distributed ROM*, or a *combination of the two* (auto). This applies the syn_romstyle attribute globally to the design by adding the attribute to the module or entity. You can also specify this attribute on a single module or ROM instance. Specifying a syn_romstyle attribute globally or on a module or ROM instance with a value of: ▶ **auto** allows the synthesis tool to choose the best implementation to meet the design requirements for peformance, size, etc. ▶ **logic** causes the ROM to be implemented using the distributed ROM or PFU resources. Specifically, the logic value will implement ROM to logic (LUT4) or ROM technology primitives (e.g., ROM16X1, ROM32X1, ROM64X1 and so on). ▶ **EBR** causes the ROM to be mapped to dedicated EBR block resources. ROM address or data should be registered to map it to an EBR block. If your ROM resources are limited, for whatever reason, you can map additional ROM to registers instead of the dedicated or distributed RAM resources using this attribute. Infer ROM architectures using a CASE statement in your code. For the synthesis tool to implement a ROM, at least half of the available addresses in the CASE statement must be assigned a value. For example, consider a ROM with six address bits (64 unique addresses). The case statement for this ROM must specify values for at least 32 of the available addresses.
-output_edif <file_name.edf>	Specifies the name of the output EDIF netlist file.
-sdc <sdc_file.ldc>	Specifies a Synopsys design constraint (.ldc) file input.
-lpf <pref_file.lpf>	Specifies whether the logical preference file (.lpf) is written or is not written. 0 or False = .lpf file not written 1 or True = .lpf file is written The default value is 1 or True.

Table 897: SYNTHESIS Command Line Options

Option	Description
-loop_limit <value>	Specifies the iteration limits for “for” and “while” loops in the user RTL for loops that have the loop index as a variable and not a constant. The higher the loop_limit, the longer the run time. Also, for some designs, a higher loop limit may cause stack overflow during some of the optimizations during compile/synthesis. The default value is 1950. Setting a higher value may cause stack overflow during some of the optimizations during synthesis.
-logfile <file_name>	Specifies the name of the synthesis log file in ASCII format. If you do not specify a name, SYNTHESIS will output a file named synthesis.log by default.
-frequency <value>	Specifies the target frequency setting. Default frequency value is 200.0 MHz.
-max_fanout <value>	Specifies maximum global fanout limit to the entire design at the top level. Default value is 1000 fanouts.
-bram_utilization <value>	Specifies block RAM utilization target setting in percent of total vacant sites. Default is 100 percent.
-mux_style <auto | pfu_mux | L6Mux_single | L6Mux_multiple>	Specifies the MUX style setting. The -mux_style option controls the way the macrogenerator implements the multiplexer macros. Valid options are *auto*, *pfu_mux*, *L6Mux_single*, and *L6Mux_multiple*. The default value is auto, meaning that the tool looks for the best implementation for each considered macro.
-fsm_encoding_style <auto | one-hot | gray | binary>	Specifies One-Hot, Gray, or Binary style. The -fsm_encoding_style. Allows the user to determine which style is faster based on specific design implementation. Valid options are *auto*, *one-hot*, *gray*, and *binary*. The default value is auto, meaning that the tool looks for the best implementation.
-use_carry_chain <0|1>	Turns on (1) or off (0) carry chain implementation for adders. The 1 or true setting is the default.
-carry_chain_length <chain_length>	Specifies the maximum length of the carry chain.
-use_io_insertion <0|1>	Specifies the use of I/O insertion. The 1 or true setting is the default.

Table 897: SYNTHESIS Command Line Options

Option	Description
-use_io_reg <0|1>	Packs registers into I/O pad cells based on timing requirements for the target Lattice families. The value 1 enables and 0 disables (default) register packing. This applies it globally forcing the synthesis tool to pack all input, output, and I/O registers into I/O pad cells. NOTE: You can place the syn_useioff attribute on an individual register or port. When applied to a register, the synthesis tool packs the register into the pad cell, and when applied to a port, packs all registers attached to the port into the pad cell. The syn_useioff attribute can be set on a: ▶ top-level port ▶ register driving the top-level port ▶ lower-level port, only if the register is specified as part of the port declaration
-resource_sharing <0|1>	Specifies the resource sharing option. The 1 or true setting is the default.
-propagate_constants <0|1>	Prevents sequential optimization such as constant propagation, inverter push-through, and FSM extraction. The 1 or true setting is the default.
-remove_duplicate_regs <0|1>	Specifies the removal of duplicate registers. The 1 or true setting is the default.
-twr_paths <timing_path_cnt>	Specifies the number of critical paths.
-dt	Disables the hardware evaluation capability.
-ip_dir <dir_name>	Switch option used for specially licensed IP core source input. Specifies the IP installation directory that contains the decoding key for the encyrpted file. NOTE: The tool does not work with IP for MachXO devices.
-corename <core_name>	Switch option used for specially licensed IP core source input. Specifies the name of IP core which should match the correpsponding name in IPexpress.
-ertl_file <file_name>	Switch option used for specially licensed IP core source input. Specifies the name of encrypted IP RTL file.
-ifd	Sets option to dump intermediate files. If you run the tool with this option, it will dump about 20 intermediate encrypted Verilog files. If you supply Lattice with these files, they can be decrypted and analyzed for problems. This option is good to for analyzing simulation issues.
-fix_gated_clocks <fix_gated_clocks_value>	Allows you to enable/disable gated clock optimization. By default, the option is enabled.
-vh2008	Enables VHDL 2008 support.
-key <key_file>	Specifies key file to encrypted IP RTL file(s).

Examples Following are a few examples of SYNTHESIS command lines and a description of what each does.

Example 1 The following command is a simple example with Verilog and VHDL file inputs.

synthesis -a MachXO2 -d LCMXO256E -top top_name -vhd f1.vhd f2.vhd f3.vhd -ver file1.v file2.v -ngd file.ngd

Example 2 The following example illustrates the usage of a search path **-p** option for IP .ngo files or include files.

synthesis -a MachXO2 -d LCMXO256E -p c:/rel/tmp c:/dir1/dir2 -top top_name -vhd f1.vhd f2.vhd f3.vhd -ver file1.v file2.v -ngd file.ngd

Example 3 The following example shows VHDL library usage with the **-lib** option.

synthesis -a MachXO2 -d LCMXO256E -top top -lib work -vhd top.vhd -lib pck -vhd ff.vhd -ngd file.ngd

Example 4 The following example illustrates the usage of both the -hdl_param and -optimization_goal options.

synthesis -a MachXO2 -d LCMXO256E -hdl_param width 7 depth 5 -optimization_goal timing -ver test1.v.v f2.v -ngd file.ngd

Example 5 This is an example of a command line with encrypted RTL for IP designs.

synthesis -a MachXO2 -d LCMXO256E -corename file_datapath -ertl_file source/file_datapath_enc.vhd -ip_dir encryption -ngd file.ngo

Example 6 This example shows miscellaneous commands for illustrating various syntax structures.

synthesis -vhd source/ora.vhd source/top.vhdl source/anda_vhd.vhd

synthesis -ver source/anda.v source/v_top.v

synthesis -a MachXO2 -d LCMXO256E -force_gsr auto -ver top.v mid.v prim.v -vhd count.vhd

synthesis -lib ..\include_lib.v -o top.ngd -lpf top.lpf -sdc top.ldc

Running EDIF2NGD from the Command Line

The **Translate Design** process in the Diamond environment can also be run through the command line using two separate programs in the following process order: **edif2ngd** and **ngdbuild**. This topic describes how to use

EDIF2NGD from the command line. Refer to "Running NGDBUILD from the Command Line" on page 2492 to complete the Translate Design process using command line tools.

EDIF2NGD is used in netlist translation. Specifically, it is a netlist reader tool that converts a netlist (.ed*, .v, .vhd) file to an .ngo file. An .ngo file is a binary file describing the design in terms of the components and hierarchy specified in the input design file. EDIF2NGD accepts FPGA properties if they appear within the input file and passes them through to the output .ngo file. All other properties are ignored and are not written to the .ngo file.

Note

Acceptable extensions for EDIF files are .edn, .edf, and .edif.

Subjects included in this topic:

- Running EDIF2NGD
- Command Line Syntax
- EDIF2NGD Options
- Examples

Running EDIF2NDG EDIF2NGD converts your input netlist (.ed*, .v, .ngo) file into an .ngo file that is used for input for NGDBUILD.

- To run EDIF2NGD, type **edif2ngd** on the command line with, at minimum, the library name with the **-l** option, your input netlist and the name of your output Native Generic Object (.ngo) file. A sample of a typical EDIF2NGD command would be as follows:

  edif2ngd -l ep5g00 design.edn design.ngo

There are many command line options that give you control over the way EDIF2NGD processes the output file. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **edif2ngd** [**-l** <libname> [**-d** <device>] [**-ip** <ippath>] [**-ic** <ipcore>] [**-nopropwarn**] [**-cbn** <format>] [**-r**] [**-f** <command_file>] <edif_file[.**edn**]> [<outfile[.**ngo**]>] [**-mc**]

EDIF2NGD Options The table below contains descriptions of all valid options for EDIF2NGD.

Table 898:

Option	Description
-l <libname>	Specifies libraries to search when determining what library components were used to build the design. NGDBUILD uses this information to resolve the components to Lattice FPGA primitives. A list of valid library names displays when you use the **edif2ngd -h** command. These names are case sensitive. Valid libraries include the following: ▶ Platform Manager 2 ▶ Platform Manager ▶ MachXO3D ▶ MachXO3LF ▶ MachXO3L ▶ MachXO2 ▶ MachXO ▶ LatticeXP2 ▶ LatticeXP ▶ LatticeSCM ▶ LatticeSC ▶ LatticeECP3 ▶ LatticeECP2M ▶ LatticeECP2 ▶ LatticeECP ▶ LatticeEC ▶ LIFMD ▶ LIFMD ▶ ECP5UM5G ▶ ECP5UM ▶ ECP5U
-d <device>	Specifies a device name. The device name option can be used as an alternative to the **-l** <libname> option. These device names are case sensitive. An example of a valid device name would be **LFECP6E**. This is a LatticeECP device example. Device names are the first half of a valid part name. For example, for a specific LatticeECP part named LFECP6ETQFP144, **LFECP6E** is the device name and **TQFP144** is the name of the package type. For more information on part names see "Selecting Valid Part Names" on page 2504 in the topic on running the **map** tool from the command line. In Diamond, device names are shown in the device selector and also will appear in the EPIC Device Editor.

Table 898:

Option	Description
-ip <ippath>	Specifies a path to a Synplify encrypted EDIF formatted IP module to be read by EDIF2NGD. When Synplify reads encrypted RTL, it generates encrypted EDIF. This option is needed for "User Configurable IP". Used in conjunction with the **-ic** option to provide the directory and name of the IP core associated with the EDIF file.
-ic <ipcore>	Specifies a path to a Synplify encrypted EDIF formatted IP core to be read by EDIF2NGD. When Synplify reads encrypted RTL, it generates encrypted EDIF. This option is needed for "User Configurable IP". Used in conjunction with the **-ip** option to provide the directory and name of the IP EDIF file for which the core is associated.
-nopropwarn	Turns off warning messages on properties. This option is useful mostly for troubleshooting and testing a design to avoid long warning messages that you might expect after running the process.
-cbn <format>	Turns on the Consistent Bus Name Conversion property taking the values **lattice**, **synplify**, or **precision**. These are not case sensitive. When this property is set to **lattice**, it converts the "( )", "[ ]", "< >", "{ }" in the bus signal names to underscores "_". For example, the bus name **aq[i]** is converted to **aq_i_**, **bq(i)** is converted to **bq_i_**, **cq<i>** is converted to **cq_i_**, **dq{i}** is converted to **dq_i_**, and so on. If given the value **synplify** or **precision**, bus signals names will be converted to match that format. Verilog and VHDL bus signal preferences will then be honored by Synplify and Precision RTL synthesis. Verilog or VHDL bus signal preferences set for Synplify will not be honored by Precision, and vice versa. If option does not appear in the command line, conversion of bus signal names does not take place.
-r	Filters out all LOCATION properties (LOC=) from the design. This can be used when you are migrating to a different device or architecture.
-f <command_file>	Executes the command line arguments in the specified command_file. See "The –f Option" on page 2583, "Command File Example" on page 2582, and "Using the Command File" on page 2583.

Table 898:

Option	Description
<edif_file[.**edn**] | [.**edf**]>	Name of the input netlist file in EDIF 2 0 0 format. If the file has an extension, you must enter the extension as part of edif_file. If you enter a filename without an extension, EDIF2NGD looks for an .edn file with the name you specified.
<outfile[.**ngo**]>	Output design file in .ngo format. The output filename, its extension, and its location are determined in this way: ▶ No output filename specified: The output file has the same name as the input file, with an .ngo extension. ▶ Output filename specified but with no extension: EDIF2NGD appends the .ngo extension to the filename. ▶ Output filename specified with an extension other than .ngo: Produces an error message and EDIF2NGD will not run. ▶ No full pathname: The output file is placed in the directory from which you ran EDIF2NGD. If the output file already exists, it will be overwritten with the new file.

Examples Following are a few examples of EDIF2NGD command lines and a description of what each does.

Example 1 The following command converts the netlist top.edn which uses the LatticeEC library to an .ngo file.

edif2ngd -l ep5g00 top.edn mapped.ngo

Example 2 The following command converts the netlist top.edn which uses the LatticeECP library to an .ngo file but uses the **-r** option to filter out all location attributes (LOC=) from the input netlist. This filtering option is useful when migrating to a different device or architecture.

edif2ngd -l ep5g00p -r top.edn mapped.ngo

Example 3 This example shows **edif2ngd** running a file command script file that is specified with the **-f** command. The command file is named edif2ngd.command.

edif2ngd -f edif2ngd.command

For example, this command file may contain several command options pertaining to multiple input netlists.

-l LatticeEC top.edn mapped.ngo
-l LatticeEC state.edn state.ngo
-l LatticeEC rom.edn rom.ngo

Example 4 This example shows **edif2ngd** using the consistent bus notation feature (-cbn) to convert bus signal names to match that synthesis format for a LatticeSC design. This ensures that during synthesis any bus signal preferences in the design are honored by the synthesis tool.

edif2ngd -l LatticeSC my.edif my.ngo -cbn synplify

See Also ▶"Command Line Data Flow" on page 2467

- "Command Line Program Overview" on page 2464

Running NGDBUILD from the Command Line

The **Translate Design** process in the Diamond environment can also be run through the command line using two separate programs in the following process order: **edif2ngd** and **ngdbuild**. This topic describes how to use NGDBUILD from the command line. Refer to "Running EDIF2NGD from the Command Line" on page 2487 to complete the Translate Design process using command line tools.

The **ngdbuild** program translates one or more .ngo files and combines each component in the design to its equivalent Lattice FPGA primitives. Before you run **ngdbuild**, you must convert the top-level design file, and all files referenced in the top-level file, to .ngo format. This is done using the **edif2ngd** program.

Subjects included in this topic:

- Running NGDBUILD
- Command Line Syntax
- NGDBUILD Options
- Examples

Running NDGBUILD NGDBUILD converts your input Native Generic Object (.ngo) file into a Native Generic Database (.ngd) file that can be used for input for MAP.

- To run NGDBUILD, type **ngdbuild** on the command line with, at minimum, the architecture name with the **-a** option, your input .ngo file, and the name of your output .ngd file. A sample of a typical NGDBUILD command would be as follows:

  ngdbuild -a ep5g00 design.ngo design.ngd

There are many command line options that give you control over the way NGDBUILD processes the output file. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **ngdbuild -a** <architecture> {**-p** <search_path>} [**-f** <command_file>] [<ngo_file.[**ngo**]>] [<outfile.[**ngd**]>] [**-cbn**] [**-mc** | **-mc**

<module_name>] [**-assemble**] | [**-h** [<architecture>] | **-help** [<architecture>] | [**-f** <command_file>]

NGDBUILD Options The table below contains descriptions of all valid options for NGDBUILD.

Table 899: NGDBUILD Command Line Options

Option	Description
-a <architecture>	Specifies the architecture of the device to which your design will be mapped (and into which the design will eventually be programmed). When you specify the architecture, the output .ngd file is optimized for mapping into that architecture. A list of valid architectures displays when you use the **ngdbuild -h** command. These names are case sensitive. Valid architecture names include the following: ▶ Platform Manager 2 ▶ Platform Manager ▶ MachXO3D ▶ MachXO3LF ▶ MachXO3L ▶ MachXO2 ▶ MachXO ▶ LatticeXP2 ▶ LatticeXP ▶ LatticeSCM ▶ LatticeSC ▶ LatticeECP3 ▶ LatticeECP2M ▶ LatticeECP2 ▶ LatticeECP ▶ LatticeEC ▶ LIFMD ▶ LIFMD ▶ ECP5UM5G ▶ ECP5UM ▶ ECP5U For backwards compatibility, the software allows you to use older architecture names for Lattice FPGA series architectures. Older Lattice FPGA architecture names are as follows: **or5s00** (LatticeSC), **mg5g00** (LatticeXP), **ep5g00** (LatticeEC), **ep5g00p** (LatticeECP). Check for availability of these devices using **-h** or by contacting support. Also see subsection "Selecting Valid Part Names" on page 2504 in "Running MAP from the Command Line" on page 2495 topic.
-dt	Disables the hardware evaluation capability.

Table 899: NGDBUILD Command Line Options

Option	Description
-f <command_file>	Execute command line arguments in the specified command file. See See "The –f Option" on page 2583, "Command File Example" on page 2582, and "Using the Command File" on page 2583.
-h [<architecture>] or **-help** <architecture>	Displays all of the available NGDBUILD command options for mapping to the specified architecture. Values for architecture are the same as for the **-a** option. If an **ngdbuild -h** command is issued without an architecture, general help is invoked.
<ngo_file[.**ngo**] >	Name of the input top-level design file in .ngo format. If you enter a filename with no extension, **ngdbuild** looks for an .ngo file with the name you specify. If you specify a filename with an extension other than .ngo, you get an error message and **ngdbuild** will not run.
<outfile[.**ngd**]>	Output design file in .ngd format. The output filename, its extension, and its location are determined in this way: ▶ No output filename specified: The output file has the same name as the input file, with an .ngd extension. ▶ Output filename specified but with no extension: EDIF2NGD appends the .ngd extension to the filename. ▶ Output filename specified with an extension other than .ngd. Produces an error message and NGDBUILD will not run. If the output file already exists, it will be overwritten with the new file.
-d <device_name>	Specifies the device type for the architecture.
-p <search_path>	Adds the specified search_path to the list of directories to search when resolving file references (i.e., files specified in the design with a FILE= filename attribute). You need not specify a search path if the necessary .ngo or .nmc file is in the directory containing the top-level .ngo file or if the FILE attribute in the design gives a complete path name for the file (instead of a relative path name). To specify multiple **-p** options, precede each with **-p**; or combine multiple search_path specifiers after one **-p**. The syntax for combining multiple search_path specifiers for a given macro search path is as follows where dir is the directory search path: **-p** dir1;dir2;dir3 (for Windows) **-p** dir1:dir2:dir3 (for UNIX)

Examples Following are a few examples of NGDBUILD command lines and a description of what each does.

Example 1 The following command translates the input netlist reader file named mapped.ngo into a Native Generic Database file named mapped.ngd.

ngdbuild -a LatticeEC mapped.ngo mapped.ngd

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Running MAP from the Command Line

The **Map Design** process in the Diamond environment can also be run through the command line using the **map** program. The **map** program takes an input generic database (.ngd) file and converts this design represented as a network of device-independent components (e.g., gates and flip-flops) into a network of device-specific components (e.g., PFUs, PFFs, and EBRs) or configurable logic blocks in the form of a Native Circuit Description (.ncd) file.

Subjects included in this topic:

- Running MAP
- Command Line Syntax
- MAP Options
- Examples

Running MAP MAP uses the generic database (.ngd) file that was the output of the **Translate Design** process (**edif2ngd** and **ngdbuild** programs) and outputs a mapped Native Circuit Description (.ncd) file and a preference file which is automatically generated if you do not specify one. The output preference file takes the name of the output NCD file.

- To run MAP, type **map** on the command line with, at minimum, the required options to describe your target technology (i.e., architecture, device, package, and peformance grade), the input .ngd along with the input .lpf file. The output .ncd file specified by the **-o** option. A sample of a typical MAP command would be as follows:

  map -p LFEC20E -t FPBGA484 -s 3 input.ngd -o output.ncd input.lpf -pr output.prf

Note

The **-a** (architecture) option is not necessary when you supply the part number with the **-p** option. There is also no need to specify the preference file here, but if you do, it must be specified after the input .ngd file name. The preference file automatically takes the name "**output**" in this case, which is the name given to the output .ncd file. If the output file was not specified with the **-o** option as shown in the above case, **map** would place a file named input.ncd into the current working directory, taking the name of the input file. If you specify the input.lpf file and it is not there, map will error out.

There are many command line options that give you control over the way MAP processes the output file. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **map** [[**-a** <architecture> **-p** <device> **-t** <pkgname> **-s** <performance> **-hier -u -m -noinferGSR -ioreg** <packing> **-pe -pr** <oprffile[**.prf**]> **-fc -c** [<packfactor>:0-100] **-retime** [<option=value:+] [**-tdm** <pref_file> **-split_node -td_pack** [<infile[**.ngd**]>] [**-o** <outfile[**.ncd**]>] [<pref_file[**.prf**]>] [<logpref_file[**.lpf**]>] | [**-h** [<architecture>] | **-help** [<architecture>] | [**-f** <command_file>]]

MAP Options The table below contains descriptions of all valid options for MAP.

Table 900: MAP Command Line Options

Option	Description
-a <architecture>	Specifies the architecture of the device to which you will map the design. The **-a** overrides the architecture specified in the input NGD file, if any. Note that this is unnecessary to specify if you use the **-p** option. A list of valid architectures displays when you use the **map -h** command. These names are case sensitive. Valid architecture names include the following: ▶ Platform Manager 2 ▶ Platform Manager ▶ MachXO3D ▶ MachXO3LF ▶ MachXO3L ▶ MachXO2 ▶ MachXO ▶ LatticeXP2 ▶ LatticeXP ▶ LatticeSCM ▶ LatticeSC ▶ LatticeECP3 ▶ LatticeECP2M ▶ LatticeECP2 ▶ LatticeECP ▶ LatticeEC ▶ LIFMD ▶ LIFMD ▶ ECP5UM5G ▶ ECP5UM ▶ ECP5U
	Note: For backwards compatibility, the software allows you to use older architecture names for Lattice FPGA series architectures. Older Lattice FPGA architecture names are as follows: **or5s00** (LatticeSC), **mg5g00** (LatticeXP), **ep5g00** (LatticeEC), **ep5g00p** (LatticeECP)**ep5m00** (LatticeECP2/M), **mj5g00** (MachXO). Check for availability of these devices using **-h** or by contacting support. Also see "Selecting Valid Part Names" on page 2504.

Table 900: MAP Command Line Options

Option	Description
-c [<packfactor>:0-100]	Allows you to set a packing factor to map to a given percentage of PFU slices in your design (1-100 percent). The value range is 0-100 percent (100 = minimum packing; 0 = maximum density). This option has great control of the packing density. It sets the relative packing density of available slices within a device in terms of a percentage of the available slices in the device. If not specified, mapping will be performed to a specific density of 97 percent. The result will be a less dense packing, depending on the size of the design relative to the number of available slices in the device. If the design is large compared with the number of available slices in the device, the mapper will make a reasonable effort to pack the design so that it fits in the device. When you specify a density percent value, the mapper attempts to pack the device to that density. The “0” setting results in the densest mapping. If the design is large compared with the target density, the mapper will make an aggressive packing effort to meet your target. However, this may adversely impact the design's f_{MAX} performance.
-f <command_file>	Executes the command line arguments in the specified command_file. See “The –f Option” on page 2583, “Command File Example” on page 2582, and “Using the Command File” on page 2583.
-h [<architecture>] or **-help** [<architecture>]	Displays all of the available MAP command options for mapping to the specified architecture. Values for architecture are the same as for the **-a** option. If an **map -h** command is issued without an architecture, general help is invoked.
-hier	Use the **-hier** option to designate hierarchical mapping instead of flat mapping. With the option designated, MAP will not pack components from different top level modules into the same PFU. This affords PAR better placement choices through improved logic grouping. Keep the following in mind when using the **-hier** option: The hierarchical mapping option will override signal sharing packing of inputs or outputs. SWLs or direct connections have higher priority than **-hier** option. The default is flat mapping, same as in previous releases. In addition, the COMP= properties will now have hierarchical paths appended to their data strings. This allows the multiple instantiation of blocks with COMP= properties to be unique and separate.

Table 900: MAP Command Line Options

Option	Description
<infile[.**ngd**]>	The design file (in .ngd format) to be mapped. You must enter an input design, but the .ngd extension is optional.
-ioreg <packing>	Direct I/O register packing. Valid packing options are **i**, **o**, **b**, and **n**. The packing options function as follows: ▶ The **i** option packs input registers only. ▶ The **o** option packs output register only. ▶ The **b** option packs both input and output registers whenever possible. ▶ The **n** option does not pack I/O registers but uses PFF/PFU resources only. This is a global option and overrides the register types in the Native Generic Database (NGD) file. Use USE DIN or USE DOUT preferences in your LPF file to override the **-ioreg** option. *This option is not applicable to MachXO devices*.
-lp <logpref_file[.lpf]>	Specifies the name of the input logical preference (.lpf) file. Backwards compatibility: If **map** does not find this file, it will error out. See Example 3 for **map** behavior in regard to the preference flow for 5.1 or higher and backwards compatibility.
-m	Allow overmapped NCDs.
-noinferGSR	Suppress GSR inferencing. By default if the input design (NGD) does not include a GSR buffer, MAP will infer one based on the signal that drives the most set/reset loads. If **-noinferGSR** is applied all GSR_NET preferences of the logical preference file (LPF) will be ignored.

Table 900: MAP Command Line Options

Option	Description
-o <outfile[**.ncd**]>	Specifies the output design file name in .ncd format. The .ncd extension is optional. If the output file already exists, it is overwritten with the new .ncd file. The output file name and its location are determined as follows: ▶ When not specified, by default the output file takes the same name as the input .ngd file, with an .ncd extension. The file is placed in the input file's directory. ▶ Output file name specified with no path (for example, cpu_dec.ncd instead of /home/designs/cpu_dec.ncd) the .ncd file is placed in the current working directory. ▶ Output file name specified with a full path (for example, / home/designs/cpu_dec.ncd) the output file is placed in the specified directory. ▶ Output file name of the physical preference (.prf) file will be the same as this output .ncd.

Table 900: MAP Command Line Options

Option	Description
-p <device>	Specifies the part number for the device. These part names are case sensitive. May be used in either of two ways: ▶ Device name only (e.g., **LFECP6E**). This is a LatticeECP device example. ▶ Part number (e.g., **LFECP6ETQFP144**) specifies both a device and a package, so you do not have to enter the **-t** option to specify the package. If you do not specify a part number using the **-p** option, MAP selects a part in this way: ▶ If the design's netlist contains properties that define a device, MAP selects the part specified by the properties. ▶ If no device properties were specified, MAP uses the default part number. ▶ Device names and part numbers are listed in the device selection in Diamond (includes part name and package name), in EPIC Device Editor, or can be obtained from technical support. Also see "Selecting Valid Part Names" on page 2504 for information on device and package selection. Examples of valid part names for various architectures include the following: ▶ **LFSC3GA15EFPBGA256** (LatticeSC) ▶ **LFXP10EFPBGA256** (LatticeXP) ▶ **LFEC20EFPBGA484** (LatticeEC) ▶ **LFECP10EPQFP208** (LatticeECP) **Note:** You can only enter a part number or device name from a device library you have installed on your system.
-pe	This option terminates the **map** process if preference errors are found.
-pr <oprffile[**.prf**]>	Optional preference file output. Prevents changing the input preference file.

Table 900: MAP Command Line Options

Option	Description
<preffile[.**prf**]>	(For ispLEVER 5.0 and below) The input physical preference file in .prf format which contains preferences written in logical preference syntax. This option remains for purposes of backwards compatibility of designs of legacy Lattice software. The previous preference flow used the PRF file to read in previous logical preference syntax allowable in the .prf file. These logical preferences used block names present in the logical netlist in their syntax and were typically LOCATE or IOBUF preferences. A physical preference file name is optional on the command line, but if one is entered it must be entered after the input NGD file name. You need not enter the .prf extension. See Example 3 for **map** behavior in regard to the preference flow for 5.1 or higher and backwards compatibility. The .mrp and .ngm files produced by a **map** run both have the same name as the output file, with the proper extension. If the .mrp or .ngm (.ldb) files already exist, they are overwritten by the new files.

Table 900: MAP Command Line Options

Option	Description
-retime or **-retime** [<option=value>]	Enables the register re-timing feature that can be used to help optimize the design to meet timing. See "Map Register Retiming" on page 598 for details of what this switch does. This option is turned off by default when **-retime** does not appear in the command line and should not be used on the initial run until you have determined whether or not it could be used to optimized your design to meet timing. If **-retime** appears in the command line, -retime TRUE is assumed and you do not have to specify the TRUE parameter. The **-retime** switch takes the following options: EFFORT Re-timing Level of Effort option. Takes integer values **1**-**6**, with **1** being the least effort and **6** the most. Default value for all devices is **4**. Syntax: **map** <map_options> **-retime** <option=value> For example, **-retime** EFFORT=2 The above **-retime** options are separated by a colon and should not contain any spaces in between them. **Note**: You may encounter error messages if you do not use a colons as a delimiter and if your parameters contain spaces. Here a few examples show the default behavior if you do not set both values: **map** <map_options> **-retime** If no options are specified as above, it is equivalent to -retime TRUE and the default option settings are used. *Not compatible with NCD Guide File*.
-s <performance>	Specifies the performance grade (toggle rate) of the part. Applicable performance grades are given in the Devices appendix. If not specified, MAP selects the performance as follows: ▶ If the designs netlist file contains a attribute that defines the device peformance, MAP selects the peformance specified by the attribute. ▶ If no device peformance attribute is specified, MAP uses the default peformance.

Table 900: MAP Command Line Options

Option	Description
-split_node	Activates timing-driven node replication. This mode replicates a LUT4 that has multiple fanout to FFs adding a LUT for each FF if the LUT belongs to the timing path. It allows packing LUT/FF in the same slice for all FFs. *Not supported by LatticeSC/M and not compatible with NCD Guide File*.
-t <pkgname>	Specifies the package (e.g., **PQFP208**, **TQFP100**, or **FPBGA484**) for the device to which you will map. These package names are case sensitive. See "Selecting Valid Part Names" on page 2504 for information on device and package selection.
-td_pack	Activates timing-driven packing of LUT/FF, FF/LUT, LUT/LUT in the same slice. *Not supported by LatticeSC/M and not compatible with NCD Guide File*.
-tdm <pref_file>	Activates timing-driven mapping for a given design flow. If you do not specify a logical preference (.lpf), by default **map** will first use the <design_name>.lpf. You can specify any other preference file you wish. If an LPF file does not exist, **map** will attempt to find a <design_name>.prf file for the backward compatibility. See example below for details. *Not compatible with NCD Guide File*.
-u	Do not remove unused logic. Note that you must use the **-u** switch in the I/O Assistant command line flow to prevent unused I/O logic from being removed since an I/O design is only partial, that is, it contains only disconnected I/O logic. This option is also especially useful when you are working with designs that have high I/O signal counts going off and coming onto your device. Unless you specify the **-u** option, the mapper will remove this unused logic from your design which will likely be undesirable.
-xref_sig	Reports signal cross references in the Map report (MRP) file. Signal cross references show where nets in the logical design were mapped in the physical design (NCD file).
-xref_sym	Reports symbol cross references in the Map report (MRP) file. Symbol cross references show where symbols in the logical design were mapped in the physical design (NCD file).

Examples Following are some examples of MAP command lines and a description of what each does.

Example 1 The following command maps an input generic database file named mapped.ngd and outputs a mapped Native Circuit Description file named mapped.ncd.and an output physical preference (.prf) file named mapped.prf. The example targets a LatticeXP device, part number lfxp3c, package PQFP208, at a -5 peformance grade.

map -a LatticeXP -p LFXP3C -t PQFP208 -s 5 mapped.ngd -o mapped.ncd my.lpf -pr mapped.prf

Example 2 The following command maps an input generic database file named mod1.ngd and outputs a mapped Native Circuit Description file named mod1.ncd by default when the output file is not specified into the current working directory. The example targets a MachXO device, part number lcmxo640c, taking default package and peformance grade because these options are not specified. The **-a** architecture is implied by the part number and is unnecessary to specify.

map -p LCMXO640C mod1.ngd

Example 3 These examples show the **map** program behavior associated with reading in logical (.lpf).

If an .lpf file is specified, **map** looks for the LPF and processes it. If the file is not found, map issues an information message that tells you that map could not open the specified preference file. Any input physical (.prf) files that are present would be ignored. Here is an example,

map -p LFEC1ETQFP100 lev1.ngd -o lev1.ncd lev1.lpf

If no .lpf or .prf file is specified, **map** looks for an LPF. If **map** does not find an .lpf file, it looks for .prf file and processes it.

map -p LFEC1ETQFP100 lev1.ngd -o lev1.ncd

Example 4 The **map** program runs with timing-driven mapping activated, using the default <design_name>.lpf file as the input preference file. If an LPF file does not exist, then **map** will look for an existing <design_name>.prf.

map -tdm

Selecting Valid Part Names Part names include both the device name and package name and can be specified in the **map** command line using the **-p** and **-t** options or by using the **-p** option with the complete part name (device name and package name with no spaces or dashes). See above examples for usage.

So, you can break down a part name into two components, the device name comes first and the package name second. For example, for a specific LatticeECP part named LFECP6ETQFP144, **LFECP6E** is the device name and **TQFP144** is the name of the package type.

- You can access the latest data sheets on www.latticesemi.com.
- In the software interface, you can view valid device and package names in the Device Selector dialog box in the Diamond environment. Please note

that in the graphical interface, these names may appear with spaces or dashes in them. Only use an alpha-numeric convention with no spaces in the command line when specifying these parts.

- Although in some cases lowercase is allowable when specifying part and package names in the command line, as a rule you should specify these names in uppercase characters. The UNIX/Linux command line is case sensitive and will not accept some part and package number arguments that use lowercase.

See Also "Command Line Data Flow" on page 2467

"Command Line Program Overview" on page 2464

Running PAR from the Command Line

The **Place & Route Design** process in the Diamond environment can also be run through the command line using the **par** program. The **par** program takes an input mapped Native Circuit Description (.ncd) file and further places and routes the design, assigning locations of physical components on the device and adding the inter-connectivity, outputting a placed and routed .ncd file.

The are special PAR Explorer (**-exp**) options available for newer Lattice FPGA architectures that help improve both placement and routing with various options for this switch described in a separate section. The Implementation Engine multi-tasking option available in UNIX is explained in detail here because the option is not available for PCs.

Subjects included in this topic:

- Running PAR
- Command Line Syntax
- General Options
- Placement Options
- Routing Options
- PAR Explorer (-exp) Options
- Examples
- PAR Multi-Tasking Option

Running PAR PAR uses your mapped Native Circuit Description (.ncd) file and associated preference (.prf) file that were the outputs of the **Map Design** process or the **map** program. With these inputs, **par** outputs a new placed-and-routed .ncd file, a PAR report (.par) file, and a PAD Specification (.pad) file that contains I/O placement information.

- To run PAR, type **par** on the command line with at minimum, the name of the input .ncd file and the desired name of the output .ncd file. While a physical preference (input.prf) file need not be specified, it is highly recommended to do so in the event there are several .prf files in the project directory. A sample of a basic PAR command would be as follows:

par input.ncd output.ncd input.prf

There are many command line options that give you control over PAR. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **par** [general_options] [placement_options] [routing_options] <infile[.**ncd**]> <outfile[.**ncd** | **.dir**]> [<preffile[.**prf**]>]

Note

All filenames without special switches must be in the order <infile> <outfile> <preffile>. Options may exist in any order.

General Options

Table 901: General PAR Command Line Options

Option	Description
-k[**p**]	Skip constructive placement. Run only optimizing placement, then enter the router. Existing routes are ripped up before routing begins. Use only on a design that has already been placed. To run re-entrant routing (also called incremental routing), use the option **-kp** to skip placement (constructive and optimizing) and begin routing, leaving the existing routing in place.
-p	Do not run placement (constructive or optimizing) route only. Existing routes are ripped up before routing begins. To run re-entrant (also called incremental) routing, use the options **-kp** to skip placement and begin routing, leaving the existing routing in place.
-r	Prevents the design from being routed. With this option, PAR outputs a placed, but not routed, Native Circuit Description (.ncd) file.
-sp <speedgrade>	Overrides the default performance grade.
-w	If the specified output file already exists, over-write the existing file (including the input file). If the specified output is a directory, overwrite files in the directory. With this option, any files generated will overwrite existing files (e.g., any .par, .pad files).
-x	Do not use timing-driven option for this PAR run.
-y	Adds the Delay Summary Report in the .par file and creates the delay file (in .dly format) at the end of the **par** run.
-pe	Issues error if Map encounters any semantic errors in the syntax of the preferences in the preference file.

Table 901: General PAR Command Line Options

Option	Description
<infile[.**ncd**]>	The design file (in .ncd format) to be placed and routed. You must enter an input design, but the .ncd extension is optional.
<outfile>	The target design file which is written after PAR is finished. ▶ If options you specify yield a single output design file, outfile may have an extension of .ncd or .dir. An .ncd extension generates an output file in .ncd format; the .dir extension directs PAR to create a directory in which to place the output file (in .ncd format). ▶ If options you specify generate more than one place and route iteration (that is, the **-n** option described above), you must specify a .dir extension; the results of each iteration are written into a separate file in the directory. These multiple files are in .ncd format. If the file or directory you specify already exists, you get an error message and the operation does not run. You can override this protection and automatically overwrite existing files by using the **-w** option.
<preffile[.**prf**]>	The preference file in .prf format containing design constraints such as timing, path specification, locking, etc. If you do not enter the name of a preference file and the current directory contains an existing preference file with the infile name and a .prf extension, PAR will use it. A preference file name is optional on the command line, but if one is entered it must be entered after the input filename. You need not enter the .prf extension.
-f <command_file>	Execute command line arguments in the specified command_file. See "The –f Option" on page 2583, "Command File Example" on page 2582, and "Using the Command File" on page 2583.

Placement Options

Table 902: PAR Placement Command Line Options

Option	Description
-a	Automatic level skipping for constructive placement. When you use **-a**, the **-n** option controls the maximum number of iterations per level. You cannot use **-a** with **-m** or **-r** options.

Table 902: PAR Placement Command Line Options

-cs <strategy>	(LatticeECP/EC and LatticeXP) Turns on clock skew minimization with a given strategy on auto-selected non-global clock nets or selected clock nets with strategy values **1** and **2** (default). This option is the same as the **-exp** clockSkewMin=<strategy> explorer option. See ClockSkewMin explorer option for additional information. The **1** value strategy option routes from driver pin to each clock load with pre-computed delay lower bound for balance routing. This option may help when the candidate clock net has a very small span, for example, when a bounding box (BBOX) span is within two to three PLCs, it will have a smaller skew. If it creates a better result, then use this option instead. The **2** value strategy option routes a clock trunk first and then routes from the trunk to each clock load with pre-computed delay with lower bound for balanced routing. This option will usually create a better result in shorter time with less memory usage.
-io	Switch that is used in the I/O Assistant process flow. This switch instructs PAR to place only I/Os, PLLs, DLLs and other elements that affect board level pin-outs.
-l <1-5>	Specifies the effort level of the design from 1 (simplest designs) to 5 (most complex designs). The default is technology dependent.
-m <nodefile_name>	Multi-tasking option. Use to run multiple nodes/machines on more than one PAR job at the same time to reduce runtimes. Cannot use **-m** with **-a**. See "Using the PAR Multi-Tasking (-m) Option" on page 2519 for details on usage. If used in conjunction with the **-n** and **-stopzero** options, once PAR returns an iteration with a timing score of zero, it will finish any jobs in the queue that have been already issued, but it will stop any further jobs from being issued and run. If necessary, you can kill jobs manually, but you may risk the program unexpectedly quitting and erroring out.

Table 902: PAR Placement Command Line Options

-n <0-100>	(Cost tables). Sets the number of iterations (place and route passes) performed at the effort level specified by the **-l** option. Each iteration uses a different cost table during placement and will produce a different .ncd file. If you enter **-n 0**, PAR continues to run until solved, stopping either after the design is fully routed and all constraints are met or after completing the iteration at cost table 100 or 99 seeds. You cannot limit the maximum number of seeds using **-n 0**. If you use the **-stopzero** option along with -n <*number*>, it will stop PAR on the first iteration that returns a zero timing score or it will end processing at the <*number*> limit you specify whether or not you reach a timing score of zero. If you do not specify the **-n** option, one place and route iteration runs. If you specify a **-t** option, the iterations begin at the cost table specified by **-t**.
-s <1-100>	Save the number of best results for this run. If not used, all results are saved.
-stopzero	Stops multiple-seed router run when the timing score reaches zero while it allows you to specify a limit to the number of iterations you are running. The maximum number of seeds is set by the **-n** option, so, for example, if you apply "-n 20 -stopzero" to a PAR run, PAR will stop running after any iteration that hits a timing score of zero or it will stop after 20 iterations are run whether or not timing is reached. If -n is not used, or "**-n 0**" is used, the router will run up to 99 seeds if timing does not hit zero. See description for the **-m** option in regard to how -stopzero prevents jobs from running iterations after it has reached a timing score of zero.
-t <1-100>	Start placement at the specified cost table. Default is 1.

Routing Options

Table 903: PAR Routing Command Line Options

Option	Description
-c <number>	Run number of cost-based cleanup passes of the router.
-d <number>	Run number of delay-based cleanup passes of the router. To run delay-based cleanup passes only on designs that have been routed to completion (100% routed), use the **-e** option instead of **-d**.
-e <number>	Run number of delay-based cleanup passes of the router on completely-routed designs only.

Table 903: PAR Routing Command Line Options

Option	Description
-i <1-1000>	Run a maximum number of passes, stopping earlier only if the routing goes to 100 percent completion and all constraints are met. If not used, runs to completion or until router decides it cannot complete.
-u	Use this router cost table entry. Valid range varies by device. The default is 1.

PAR Explorer Options The PAR Explorer (**-exp**) options are specialized options that help improve both the placement and routing of a given design. The options are available for most Lattice FPGA designs. Generally, most explorer options are not available for the LatticeSC/M family unless stated otherwise. Use the **-h** option with the architecture argument in the PAR command line to determine what options are available per device family.

Table 904: PAR Explorer Command Line Options

Option	Description
-exp [<option=value>:+]	Explorer string to turn on or off special placement and routing options. Options and values are case-insensitive. You can also specify multiple options using a colon to indicate additional options in the following manner: Syntax: **-exp** [<option=value>:+] Example: **par -exp** option1=value:option2=value2 design.ncd *All valid -exp values are listed in the table rows below in alphabetical order*.
Valid -exp Values	

Table 904: PAR Explorer Command Line Options

ClockSkewMin=<strategy>	CLOCKSKEWMIN Clock Skew Minimization option. This is the same as the **-cs** option. The option is used minimize your clock skew on selected clock nets that warrant further optimization. Options are **1** and **2** (default) and are referred to as "strategies." You turn this option on after you identified good candidate clock nets that have excessive skew in the TRACE timing report. This option is enabled by a program called CSM that performs optimization. The syntax of this command is as follows: **par -exp** clockSkewMin=<strategy> A indicator of a good candidate would be any FREQUENCY preference that has substantial skew on its driving clock net. This will usually occur with nets that cannot be assigned to primary or secondary routing resources by default. Check such nets to see if they are gated clocks. CSM will have no effect on gated clocks since they create interclock domains. If the candidate skewed net named "clk_c" is a pure clock net (one level only), then add the following two preferences to PRF file for rerunning PAR: PROHIBIT PRIMARY NET "clk_c" ; PROHIBIT SECONDARY NET "clk_c" ; The **clockSkewMin=1** strategy option routes from driver pin to each clock load with pre-computed delay lower bound for balance routing. This option may help when the candidate clock net has a very small span, for example, when a bounding box (BBOX) span is within two to three PLCs, it will have a smaller skew. If it creates a better result, then use this option instead. The **clockSkewMin=2** strategy option routes a clock trunk first and then routes from the trunk to each clock load with pre-computed delay with lower bound for balanced routing. This option will usually create a better result in shorter time with less memory usage. See the example and "Clock Skew Minimization" on page 610 for further details on available application notes on this feature.
M5GRipUp=(ON | OFF)	M5GRIPUP M5G-specific rip-up option. Options are **On** (default) and **Off**. Turning this option **On** will typically improve design routability.

Table 904: PAR Explorer Command Line Options

parCB=(ON | OFF)	PARCB Clock boosting option. This option forces the router to automatically insert extra wires to a Primary or Secondary route clock path to help compensate for hold time violations. Compatible families: ECP5, LatticeECP/EC, LatticeECP2/M, LatticeECP3, LatticeXP, LatticeXP2, MachXO, MACHXO2, MachXO3D, and MachXO3L families. **Note**: PARCB has shown the most benefit on design benchmarks targeted to LatticeECP3 device families. Options are **ON** and **OFF**, with **OFF** being the default condition. The syntax of this command is as follows: **par -exp** parCB=ON my.ncd Clock boosting using hardware delay taps are available for LatticeSC/M.
parPathBased=(ON | OFF)	PARPATHBASED Path-based placement option. Path-based timing driven placement will yield better performance and more predictable results in many cases. Options are **Off** (default) and **On**. The syntax of this command is as follows: **par -exp** parPathBased=ON my.ncd
parClkNetWeight=<value>	PARCLKNETWEIGHT Performance-driven clock selection option. Options are **100**, **150**, and **200** and so on. You can set this parameter to select the weight factor between fanouts and performance constraints. When using 150, the weight for critical net will be fanout * 1.5, so the critical net will be more likely selected as primary clock. if using 200, fanout * 2.0. If set value < 100, it is equivalent to 100. The syntax of this command is as follows: **par -exp** parClkNetWeight=150 my.ncd
parCDP=[0|1|Auto]	Enable the congestion-driven placement (CDP) algorithm. CDP is compatible with all Lattice FPGA device families; however, most benefit has been demonstrated with benchmarks targeted to ECP5, LatticeECP2/M, LatticeECP3, and LatticeXP2 device families. Usage: **par -exp** parCDP=[0|1|Auto] where: **0** (default) - disables congestion-driven placement. **1** - enables congestion-driven placement. **Auto** – allows par to enable CDP automatically. See parUseNBR and parCDR options for more congestion-driven options.

Table 904: PAR Explorer Command Line Options

parCDR=[0|1|Auto]	Enable the congestion-driven router (CDR) algorithm. Congestion-driven options like parCDR and parCDP can improve performance given a design with multiple congestion "hotspots." The **Layer > Congestion** option of the Design Planner Floorplan View can help visualize routing congestion. Large congested areas may prevent the options from finding a successful solution. CDR is compatible with all Lattice FPGA device families however most benefit has been demonstrated with benchmarks targeted to ECP5, LatticeECP2/M,LatticeECP3, and LatticeXP2 device families. Usage: **par -exp** parCDR=[0|1|Auto] where: **0** (default)– disables congestion-driven router. **1** – enables congestion-driven router. **Auto** – allows par to enable CDR automatically. See parUseNBR and parCDP options for more congestion-driven options.

Table 904: PAR Explorer Command Line Options

parHold=(0 | 1 | 2)	PARHOLD Additional hold time correction option. By default, hold time correction will be performed automatically if setup time error is 0. This additional option will overwrite the default setting. If parHold=0, hold time correction will be skipped. If parHold=1, hold time correction will be run even if setup error exists. If parHold=2, hold time correction will run with extra effort. This option forces the router to automatically insert extra wires to compensate for the hold time violation. When turned on, the router will automatically try to resolve any hold time violations on all registers with preferences placed on them including input I/O to registers, register to register, and register to output I/O. This will also cause negative impact on your setup time. If the setup time is more design critical, you should turn this switch off. For backwards compatibility for earlier releases of legacy Lattice software, you can use the MACHXO_HOLD value. The syntax of this command is as follows: **par -exp** parHold=1 my.ncd Other available options in auto hold time correction include: parHold=1 (turns on auto hold time correction) parHold=0 (turns off auto hold time correction) parHold=2 (run hold time correction with extra effort) parHoldSpeedGrade= M (minimum) **Note**: When hold time correction is called, the default performance grade used in hold time correction is always -M or the minimum performance grade allowable. Examples: par -w mapped.ncd output.ncd design.prf -exp parHold=1 par -p -w placed.ncd output.ncd design.prf -exp parHold=1 par -p -w placed.ncd output.ncd design.prf -exp parHold=1 par -p -w placed.ncd output.ncd design.prf -exp parHold=1:parHoldIter=5:parHoldLimit=500:parHoldSpeedGrade=6 This correction option can be used in two different manners, inside the standard PAR flow or as a standalone program. Inside the standard PAR flow, auto hold time correction is implemented as an independent router task, so it can be used together with other router tasks such as a cleanup task.

Table 904: PAR Explorer Command Line Options

parHold=(0 | 1 | 2) (*cont.*)	PARHOLD (*cont.)* Using Auto Hold Time Correction as a Standalone Program The correction option can also be called as a standalone program. If you have a previously routed NCD and you want to check if there are any hold time violations, it is useful to use the standalone auto hold time correction program. The input to the standalone program is a completely routed NCD file. The standalone program will check whether the input NCD has hold time violations in a user-specified performance grade, and will correct the violations if there are any. Using this approach, the performance grade used by the standalone program does not have to match the performance grade used in PAR. Usage Examples: 1. Run a standard PAR flow with “parHold=0” to get a completely routed NCD file, as shown below: par -w -i 6 mapped.ncd routed.ncd design.prf **-exp parHold=0** par -p –sp 5 -w -i 6 placed.ncd routed.ncd design.prf **-exp parHold=0** 2. Run the standalone auto hold time correction program with a completely routed NCD as the input as shown below: par **-kp** –w routed.ncd output.ncd design.prf -exp **parHold=1**: **parHoldOnly=1** par **-kp** –w routed.ncd output.ncd design.prf -exp **parHold=1**: **parHoldOnly=1** par **-kp** –w routed.ncd output.ncd design.prf -exp **parHold=1**: **parHoldOnly=1**:parHoldIter=5:parHoldLimit=500: parHoldSpeedGrade=6 The options in bold font in the examples above are required to run the standalone program. If no performance grade is specified, the standalone program will default to the -M or minimum performance grade. The **-kp** option is used explicitly for re-entrant routing, instructing PAR to skip placement and leave existing routing in place. There are situations where it is desirable to run the standalone auto hold time correction flow. For example, this flow is advantageous if you have a previously routed NCD and you would like to check if it has contains any hold time violations. The standalone program flow will check if the input routed NCD has hold time violations in the selected performance grade and will correct those violations.
parHoldIter=<value>	PARHOLDITER Specifies the number of automatic hold time correction iterations. If not specified with the -exp parHold PAR option, the default value is 4. For example: par -kp –sp M –w routed.ncd output.ncd design.prf -exp **parHold=1**:parHoldOnly=1:**parHoldIter=5:**parHoldLimit=500

Table 904: PAR Explorer Command Line Options

parHoldLimit=<value>	PARHOLDLIMIT This option allows you to set a limit on the number of hold time violations to be processed by the auto hold time correction option parHold. If there are more hold time violations in a design than the set limit, the auto hold time correction will disable itself and cease attempting to correct any violations and a warning message will appear. At this juncture, you should examine your design to see if you can avoid these violations. If there are too many hold time violations in a design, it may take longer runtime to correct these violations. This option provides a way to limit the number of potential violations to be processed. The default value is 250. The syntax of this command is as follows: par -exp **parHold=1**:**parHoldlimit=125** my.ncd
parHoldOnly=<1l0>	PARHOLDONLY Required option to run standalone PAR auto hold time correction flow with -exp parHold PAR option. See usage for parHold option. By default, parHold value is “0” or is turned off. When turned on, “1”, the option indicates that this PAR run is for auto hold time correction only. For example: par **-kp** –w routed.ncd output.ncd design.prf -exp **parHold=1**: **parHoldOnly=1**
parHoldSpeedGrade= <value>	PARHOLDSPEEDGRADE This option sets the performance grade to be used in auto hold time correction with the -exp parHold option. This performance grade can be different from the performance grade used in PAR. Default value is parHoldSpeedGrade =M. An example of how this may look on the command line is as follows: par -kp –sp M –w routed.ncd output.ncd design.prf -exp **parHold=1**:parHoldOnly=1:parHoldIter=6:parHoldLimit=300:parHoldSpeedGrade=5
parUseNBR=[0|1]	PARUSENBR NBR Router or Negotiation-based routing option. Supports all FPGA device families except LatticeXP and MachXO. When turned on, an alternate routing engine from the traditional Rip-up-based routing selection (RBR) is used. This involves an iterative routing algorithm that routes connections to achieve minimum delay cost. It does so by computing the demand on each routing resource and applying cost values per node. It will complete when an optimal solution is arrived at or the number of iterations is reached.

Table 904: PAR Explorer Command Line Options

timing_effort=<value>	TIMING_EFFORT This option makes the maximum effort to achieve a placement and routing solution. It is only intended to be used when you have run older legacy Lattice software on a design (ispLEVER 5.1 or earlier), then run the newer version on it, and the maximum frequency (f_{MAX}) drops as a result. The syntax of this command is as follows: par -exp timing_effort=5 my.ncd The level of effort can only be set to 5.
parASE=[0|1]	PARASE This option is used for multi-seed runs of Place & Route. It reduces overall run time by terminating placement seeds that are not producing better results. The Run Time Reduction option uses checkpoints for comparing the result of the current placement seed with the results already obtained. During the optimization phase, if clock boosting and/or automatic hold time correction have been enabled, the PAR engine will also check whether one or both options are needed and optimize the routing accordingly. A placement seed that is not producing better results will automatically be terminated.

Examples Following are a few examples of PAR command lines and a description of what each does.

Example 1 The following command places and routes the design in the file input.ncd and writes the placed and routed design to output.ncd.

par input.ncd output.ncd

Example 2 The following command skips the placement phase and preserves all routing information without locking it (re-entrant routing). After that, the command instructs PAR to run up to 999 passes of the router or stops upon completion and conformance to timing preferences found in the pref.prf file. Finally, it runs three delay-based cleanup router passes. If the design is already completely routed, the effect of this command is to just run three delay-based cleanup passes.

par -kp -i 999 -c 0 -d 3 input.ncd output.ncd pref.prf

Example 3 The following command runs 20 place and route iterations at effort Level 3. The iterations begin at cost table entry 5. Only the best 3 output design files are saved. The output design files (in .ncd format) are placed into a directory called results.dir.

par -n 20 -l 3 -t 5 -s 3 input.ncd results.dir pref.prf

Example 4 The following command copies the input design to the output design using the **-pr** option. The placement and routing phases are both skipped completely. You generate a delay file using the **-y** option, which creates a delay file that allows the user to check the delay times in the design without having PAR change any of the design's placement or routing.

par -pr -y input.ncd output.ncd

Example 5 The following example consists of two command lines. Suppose that you have determined the best level for the design to be level 3. If you wanted PAR to save the best three results running at level 3 with 20 place-and-route iterations, you would enter this command:

par -l 3 -n 20 -w -s 3 input.ncd output.dir input.prf

Now, if you wanted to run two passes of cost-based and delay-based cleanup on the three designs saved (without running placement), you would enter this command for each design:

par -kp -i 0 -c 2 -d 2 input.ncd output.ncd input.prf

Example 6 The following command below runs optimized placement and routing. The constructive placement results from a previous run are kept and refined with placement optimization. The router starts from scratch.

par -k input.ncd output.ncd input.prf pref.prf

Example 7 The command below allows re-entrant routing. Use this command when your design is only partially routed and you want to complete it. Placement and placement optimization are skipped. In this case up to 30 router passes are run (you could run up to 999).

par -kp -i 30 input.ncd output.ncd

Example 8 (Lattice FPGAs only) This is an example of **par** using the **-io** switch to generate .ncd files that contain only I/O for viewing in the PAD Specification file for adjustment of LOCATE preferences for optimal I/O placement. You can display I/O placement assignments in Diamond's Spreadsheet view and choosing **View > Display IO Placement**.

par -io -w lev1bist.ncd lev1bist_io.ncd

Example 9 The command below uses the clock skew minimization PAR Explorer (**-exp**) option (equivalent to the **-cs** option) to reduce clock skew on a clock net which had been identified as having excessive skew and fit the rest of the candidate criteria. See the option description for -exp clockSkewMin for details on usage.

par -sp 5 -i 6 -exp clockSkewMin=2 -w EC20.ncd EC20_s2.ncd par.prf

or,

par -sp 5 -i 6 -cs 2 -w EC20.ncd EC20_s2.ncd par.prf

Note that the following unique preferences must be given to select exactly which net to which you will apply clock skew minimization.

```
PROHIBIT PRIMARY NET <net_name> ;
PROHIBIT SECONDARY NET <net_name> ;
```

The above two preferences must be specified simultaneously for a particular net to be considered for clock skew minimization. And this net should have load number not exceeding 300. However, this value can be adjusted by using the -exp CLOCK_BALANCE_MAXLOADS=<value> option, if appropriate. For example:

```
# specify clock net "clk_c" to be selected for CSM
PROHIBIT PRIMARY NET "clk_c";
PROHIBIT SECONDARY NET "clk_c" ;
```

Note

If no **-exp clockSkewMin** or **-cs** option is given in the command line, the net “clk_c” specified above will be routed using regular routing resources by regular maze router as current flow works.

Using the PAR Multi-Tasking (-m) Option This section provides information about environment setup, node list file creation, and step-by-step instructions for running the PAR Multi-tasking (**-m**) option from the command line. The PAR **-m** option allows you to use multiple machines (nodes) that are networked together for a multi-run PAR job, significantly reducing the total amount of time for completion. Before the multi-tasking option was developed, PAR could only run multiple jobs in a linear or serial fashion. The total time required to complete PAR was equal to the amount of time it took for each of the PAR jobs to run.

For example, the PAR command:

par -l 5 -n 10 -i 10 -c 1 mydesign.ncd output.dir

tells PAR to run 10 place and route passes (**-n 10**) at effort level 5 (**-l 5**), a maximum of 10 router passes (**-i 10**), and one cost-based cleanup pass. It runs each of the 10 jobs consecutively, generating an output .ncd file for each job, i.e., output.dir/ 5_1.ncd, output.dir/5_2.ncd, etc. If each job takes approximately one hour, then the run takes approximately 10 hours.

Suppose, however, that you have five nodes available. The PAR Multi-tasking option allows you to use all five nodes at the same time, dramatically reducing the time required for all ten jobs.

To run the PAR multi-tasking option from the command line:

1. First generate a file containing a list of the node names, one per line as in the following example:

```
# This file contains a profile node listing for a PAR multi
# tasking job.
[machine1]
SYSTEM = linux
CORENUM = 2
[machine2]
SYSTEM = linux
CORENUM = 2
Env = /home/user/setup_multipar.lin
```

```
Workdir = /home/user/myworkdir
```

You must use the format above for the node list file and fill in all required parameters. Parameters are case insensitive.The node or machine names are given in square brackets on a single line.

The **System** parameter can take linux or pc values depending upon your platform. However, the PC value cannot be used with Linux because it is not possible to create a multiple computer farm with PCs. **Corenum** refers to the number of CPU cores available. Setting it to zero will disable the node from being used. Setting it to a greater number than the actual number of CPUs will cause PAR to run jobs on the same CPU lengthening the runtime.

The **Env** parameter refers to a remote environment setup file to be executed before PAR is started on the remote machine. This is optional. If the remote machine is already configured with the proper environment, this line can be omitted. To test to see if the remote environment is responsive to PAR commands, run the following:

```
ssh <remote_machine> par <par_option>
```

See the System Requirements section for details on this parameter.

Workdir is the absolute path to the physical working directory location on the remote machine where PAR should be run. This item is also optional. If an account automatically changes to the proper directory after login, this line can be omitted. To test the remote directory, run the following,

```
ssh <remote_machine> ls <ncd_file>
```

If the design can be found then the current directory is already available.

2. Now run the job from the command line as follows:

```
par -m nodefile_name -l 5 -n 10 -i 10 -c l mydesign.ncd
output.dir
```

This runs the following jobs on the nodes specified.

```
NODE1: par -l 5 -i 10 -cl mydesign.ncd output.dir/5_1.ncd
NODE2: par -l 5 -i 10 -cl mydesign.ncd output.dir/5_2.ncd
NODE3: par -l 5 -i 10 -cl mydesign.ncd output.dir/5_3.ncd
NODE4: par -l 5 -i 10 -cl mydesign.ncd output.dir/5_4.ncd
NODE5: par -l 5 -i 10 -cl mydesign.ncd output.dir/5_5.ncd
```

As the jobs finish, the remaining jobs start on the nodes until all 10 jobs are complete. Since each job takes approximately one hour, all 10 jobs will complete in approximately two hours.

You cannot use the **-a** option (automatic level skipping) with the **-m** nodefile option (the Implementation Engine).

Note

If you attempt to use the multi-tasking option and you have specified only one placement iteration, PAR will disregard the **-m** option from the command and run the job in normal PAR mode. In this case you will see the following message:

```
WARNING - par: Multi task par not needed for this job. -m
switch will be ignored.
```

System Requirements **ssh** must be located through the PATH variable. On Linux, the utility program's secure shell (**ssh**) and secure shell daemon (**sshd**) are used to spawn and listen for the job requests.

The executables required on the machines defined in the node list file are as follows:

- /bin/sh
- par (must be located through the PATH variable)

Required environment variable on local and remote machines are as follows:

- FOUNDRY (points at FOUNDRY directory structure must be a path accessible to both the machine from which the Implementation Engine is run and the node)
- LM_LICENSE_FILE (points to the security license server nodes)
- LD_LIBRARY_PATH (supports par path for shared libraries must be a path accessible to both the machine from which the Implementation Engine is run and the node)

To determine if everything is set up correctly, you can run the **ssh** command to the nodes to be used.

Type the following:

```
ssh <machine_name> /bin/sh -c par
```

If you get the usage message back on your screen, everything is set correctly. Note that depending upon your setup, this check my not work even though your status is fine.

If you have to set up your remote environment with the proper environment variables, you must create a remote shell environment setup file. An example of an ASCII file used to setup the remote shell environment would be as follows for ksh users:

```
export FOUNDRY=/home/rel/rtf7_1b.28/
export PATH=$FOUNDRY/bin/lin:$PATH
export LD_LIBRARY_PATH=$FOUNDRY/bin/lin:$LD_LIBRARY_PATH
```

For csh users, you would use the setenv command.

Screen Output When PAR is running multiple jobs and is not in multi-tasking mode, output from PAR is displayed on the screen as the jobs run. When PAR is running multiple jobs in multi-tasking mode, you only see information regarding the current status of the feature.

For example, when the job above is executed, the following screen output would be generated:

```
Starting job 5_1 on node NODE1
Starting job 5_2 on node NODE2
Starting job 5_3 on node NODE3
Starting job 5_4 on node NODE4
Starting job 5_5 on node NODE5
```

When one of the jobs finishes, this message will appear:

```
Finished job 5_3 on node NODE3
```

These messages continue until there are no jobs left to run.

See Also ▶"Running Multiple PAR Jobs in Parallel" on page 635

- "Command Line Data Flow" on page 2467
- "Command Line Program Overview" on page 2464

Running MPARTRCE from the Command Line

MPARTRCE is a command line utility that manages the place-and-route and timing wizard programs, PAR and TRCE to derive the best possible PAR run based on performance criteria and generating an optimal placed-and-routed NCD file. For details on PAR and TRACE options see the related online help topics for running each program from the command line.

For more details on the MPARTRCE methodology, performance score formula, applying weighted preferences, and other more advanced subject areas, see "Using MPARTRCE to Optimize PAR Iterations" on page 613 and "Specifying Preferences for MPARTRCE" on page 615.

Subjects included in this topic:

- How MPARTRCE Works
- Running MPARTRCE
- Command Line Syntax
- MPARTRCE Options
- Examples
- MPARTRCE Reports

How MPARTRCE Works To perform its task, MPARTRCE first reads in constraints and any associated weights and penalties from the PRF file. Then, it runs PAR, TRCE, and an algorithm for selection multiple times. Using the TRACE report (.twr), it uses values to determine frequency and performance scores and creates a report in .CSV format.

In most cases, MPARTRCE picks the best implementation that meets the user preferences using the default files and settings. By default, MPARTRCE uses the same preference file that PAR uses. It extracts the timing preferences of interest (FREQUENCY, PERIOD, CLOCK_TO_OUT and INPUT_SETUP) from the preference file for use in performance score computation. The weights and penalties are read from an initialization file, if one exists or from the hard- coded factory defaults.

If MPARTRCE is interrupted after any iteration N, the results and reports correspond to the best implementation thus far. The outputs would be equivalent to what would have obtained if MPARTRCE had run with N iterations.

Running MPARTRCE MPARTRCE uses your input mapped Native Circuit Description (<design>_map.ncd) file and associated preference (.prf) file from PAR or TRCE to generate multiple new NCD file seeds to select from that best fit your performance criteria. These NCD files are automatically output to a results directory called <design>.dir that is placed in your project directory. A log file and report file is automatically generated to help you in your selection process.

- To run MPARTRCE, type **mpartrce** on the command line with, at minimum, the name of your design. A sample of a typical MPARTRCE command might be as follows:

  mpartrce mydesign

There are several commands options that do not need to be specified due to the default behavior of MPARTRCE, so please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **mpartrce** [**-p** <file>] [**-f** <command_file>] [**-tf** <file>] [**-log** <file>] [**-rem**] [**-o** <reportfile>] **-pr** <preffile.prf> [**-tpr** <preffile.prf>] <design[**.ncd**]> <designout[**.ncd**]>

MPARTRCE Options The table below contains descriptions of all valid options for MPARTRCE.

Table 905: MPARTRCE Command Line Options

Option	Description
-p <file**>**	PAR command file. This is the same as the .p2t file used by PAR program. If this option is not specified, by default the <design>.p2t is used.
-f <command_file>	Execute command line arguments in the specified command file. See "The –f Option" on page 2583, "Command File Example" on page 2582, and "Using the Command File" on page 2583.

Table 905: MPARTRCE Command Line Options

Option	Description
-tf <file>	TRCE command file. This file is passed to the **trce** command with a **f** option. Refer to "Running TRACE from the Command Line" on page 2526 for details on the **f** file format.
-log <file>	Specifies a Log file name. The log collects the timing results from the .twr file for all the iterations. This option is mandatory unless you run the abbreviated command for this tool, that is, you only enter the **mpartrce** command with no other arguments. When you specify just the **mpartrce** command without arguments, a log file named <design>.log is created.
-rem	Removes the previously generated design.dir output directory before it processes new output. This directory contains multiple NCD files from PAR based on performance criteria and is generated automatically.
-o <reportfile>	Specifies a report file in CSV format. This file contains the summary report of constraints used for result selection, PSi, PA and timing score for each iteration. This option is mandatory unless you run the abbreviated command for this tool, that is, you only enter the **mpartrce** command with no other arguments. When you specify just the **mpartrce** command without arguments, a log file named <design>.csv is created.
-pr <preffile.prf>	PAR preference file. This is the same as <design>.prf file. If this option is not specified, by default <design>.prf is used.
-tpr <preffile.prf>	TRACE preference file, that is, the preference file used for the **trce** program and result selection. The preference file with the **-tpr** option can be the same as the PAR preference file or a separate **trce**-only preference file (usually, created from PAR preference file). If this option is not specified, by default <design>.prf is used for the TRACE preference file.
<design[**.ncd**]>	Design file on which to run PAR. This file is the same as <design>_map.ncd.
<designout[**.ncd**]>	Output Native Circuit Description (.ncd) file for multiple PAR seeds. The target design file is written out to an auto-generated directory within your project directory after PAR is finished. If this option is not specified, by default a the output directory is called <design>.dir. This directory contains your output NCD and TRACE report files for multiple iterations. The output directory also contains your log and your MPARTRCE summary report file in .CSV file format.
-h	Displays help on available options and syntax.

Examples The following include a few examples of MPARTRCE command lines and a description of what each does.

Example 1 The **-rem** option is used here to remove a prior output directory that was generated for MPARTRCE. The output directory will be called counter_map.dir as it takes the name of the design file by default. The **-p** option specifies the use of a PAR command file. Log and report files will be regenerated with the use of the **-log** and **-o** options, respectively with the name specified.

mpartrce -rem -p counter.p2t -pr counter.prf -log counter.log -o counter.csv counter_map.ncd counter.ncd

Example 2 The following command line example shows how to specify TRACE preference file usage with the **-tpr** option.

mpartrce -p counter.p2t -pr counter.prf -tpr counter_trc.prf -log counter.log -o counter.csv counter_map.ncd counter.ncd

Example 3 The following abbreviated command line example illustrates how to use the minimal command line argument to run **mpartrce**. To do so, you would just enter the executable with the design name argument specified. The name of your design in this case is “counter” and the placed-and-routed NCD file name would be counter.ncd.

This command will use all established input project file names (e.g., counter.prf) and output counter.log and counter.csv log and report files.

mpartrce counter

MPARTRCE Reports Below is an example of a report .CSV file that MPARTRCE generates that you can analyze.

Table 906: MPARTRCE .CSV Report File

---- MParTrce Report File ----											
	preferen ce							constrai nt	weight	penalty	
constrai nt 1	CLOCK_TO_OUT ALLPORTS 7.000000 ns "clk"				CLKPORT			7.000 ns	1	10	
constrai nt 2	FREQUENCY NET "mclk" 200.000000 MHz							200.000 MHz	20	400	
constrai nt 3	FREQUENCY PORT "clk" 200.000000 MHz							200.000 MHz	20	400	
constrai nt 4	INPUT_SETUP ALLPORTS 3.000000 ns "clk"				CLKPORT			3.000 ns	1	10	
index	iteration	Actual 1(ns)	PS 1	Actual 2(MHz)	PS 2	Actual 3(MHz)	PS 3	Actual 4(ns)	PS 4	Perf. Score	timing score

Table 906: MPARTRCE .CSV Report File

1	5	5.0	2.0	202.5	1.2	238.0	16.0	2.9	0.1	19.3	0
2	10	5.2	1.8	200.7	0.4	238.0	16.0	2.4	0.6	18.7	0
3	8	5.0	2.0	200.0	0.0	238.0	16.0	2.8	0.2	18.2	1
4	3	4.9	2.1	200.0	0.0	238.0	16.0	3.0	0.0	18.1	0
5	4	5.4	1.6	200.6	0.3	238.0	16.0	2.8	0.2	18.0	0
6	6	5.6	1.4	200.1	0.1	238.0	16.0	2.9	0.1	17.5	0
7	7	5.3	1.7	199.6	-4.0	238.0	16.0	3.0	0.0	13.7	10
8	1	6.0	1.0	189.6	-109.6	238.0	16.0	2.7	0.3	-92.4	332
9	9	5.5	1.5	189.3	-113.6	238.0	16.0	2.7	0.3	-95.9	1136
10	2	4.7	2.3	187.2	-136.8	238.0	16.0	2.3	0.7	-117.9	976

5 is the best.

See Also ▶"Using MPARTRCE to Optimize PAR Iterations" on page 613

- "Specifying Preferences for MPARTRCE" on page 615
- "Multiple PAR Iterations" on page 612

Running TRACE from the Command Line

The **MAP TRACE** and **Place & Route TRACE** processes in the Diamond environment can also be run through the command line using the **trce** program. TRACE (Timing Reporter and Circuit Evaluator) or the **trce** program can be run on designs using the placed and routed Native Circuit Description (.ncd) and associated timing preferences specified in the design's preference (.prf) file. Using these input files, **trce** provides static timing analysis and outputs a timing report file (.tw1/.twr).

TRACE checks the delays in the Native Circuit Description (.ncd) file against your timing preferences. If delays are exceeded, TRACE issues the appropriate timing error. See "Mapping" on page 590 and associated topics for more information.

Subjects included in this topic:

- Running TRCE
- Command Line Syntax
- TRCE Options
- Examples

Running TRCE TRCE uses your input mapped or placed-and-routed Native Circuit Description (.ncd) file and associated preference (.prf) file to create a TRACE Timing report.

- To run TRCE, type **trce** on the command line with, at minimum, the names of your input .ncd and .prf files to output a timing report (.twr) file. A sample of a typical TRCE command would be as follows:

  trce design.ncd design.prf

Note

The above command automatically generates the report file named design.twr which is based on the name of the .ncd file.

There are several command line options that give you control over the way TRCE generates timing reports for analysis. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **trce** [**-allprefpath**] [**-fullname**] [**-clockdomain**] [**-gt** [<percentage>]] [**-e** [<n>] **-v** [<limit>]] [**-p**] [**-sp** <peformance>] [**-sphld** <peformance>] [**-min**] [**-max**] [**-c**] [**- u** <limit>] [**-mapchkpnt** <number>] [**-hld**] [**-setuphold** [v:0,2]] [**-sethld**] [**-primetime**] [**-html**] [**-o** <report[**.twr**]>] <file_name[**.ncd**]> [<file_name[**.prf**] >] [**-f** <command_file>]

TRCE Options

The table below contains descriptions of all valid options for TRCE.

Table 907: TRCE (TRACE) Command Line Options

Option	Description
-allprefpath	Specifies that a path for each timing preference be reported even if it belongs to two or more timing preferences.
-c	Prints connections not covered by the preference file. The **-c** option should be used with the verbose timing report option.
-u	Prints paths not covered by the preference file. The **-u** option cannot be used with the **-allprefpath** command line option.
-clockdomain	Performs clock domain analysis, reporting clock ports and clock nets relationships.
-e <n>	The **-e** generates an error report file in the current working directory. If this or the **-v** option is unspecified, TRACE sends a summary report to standard output. The <n> value specifies the allowable number of paths with timing errors to be reported.
-f <command_file>	Execute command line arguments in the specified command_file. See "The –f Option" on page 2583, "Command File Example" on page 2582, and "Using the Command File" on page 2583.

Table 907: TRCE (TRACE) Command Line Options

Option	Description
<file_name[.**ncd**]>	Specifies the Native Circuit Description file.
<file_name[.**prf**]>	The .prf file containing timing constraints for the design file. If a preference file is not specified, TRACE looks for one with the same root name as the .ncd file.
-fullname	Prevents TRACE from truncating COMP names in the output report. By default, truncated names appear in report file preceded by an asterisk.
-gt <percentage>	The -gt option enables auto-generation of timing preferences if there are no timing preferences found in scan chain paths. Percentage value factors are 0-100 which represent the routing percentage relative to total delay, or the routing delay plus logic delay. The generated timing preferences will be based on this delay estimation percentage.
-hld <file_name[.**ncd**]> <**-v**> <file_name[.**prf**]>	The **-hld** option performs hold time checks on FREQUENCY/PERIOD, CLOCK_TO_OUT, INPUT_SETUP and OFFSET IN/OUT preferences and outputs a verbose timing report in your working directory. When using the **-hld** option to do a hold time check, the default peformance grade of **-M** (minimum delay) is ascribed when no **-sp** option (valid peformance grade value) is specified to override this value. For setup time checks (TRACE timing analysis which does not include the **-hld** option), the default peformance grade will be derived automatically from the .ncd file based upon device selection.
-html	Outputs an HTML-like format report.
-mapchkpnt <number>	Controls the algorithm used to estimate timing for routing after MAP and prior to placement. This estimation is used in the MAP Trace report (.tw1) file and the MAP Timing Checkpoint. ▶ If <number> is between 1 and 100, use the Estimated Logic Delay algorithm. The value of <number> represents the how much of the total path delay will be from the path's logic delay. For example, **100** creates zero routing delay for a path since the logic delay represents 100 percent of the total path delay. A number of **50** causes the route delay to be equal to the logic delay, and thus the total path delay is twice the logic delay. ▶ If <number> = **0**, then the routing delay is estimated using the Timing Design Suite which is based on routing delay seen on the set of critical paths across a suite of benchmark designs. ▶ If no –mapchkpnt switch is used, then the default behavior is **–mapchkpnt 100** which results in zero routing delay.

Table 907: TRCE (TRACE) Command Line Options

Option	Description
-max	Uses maximum delay numbers only.
-min	Uses minimum delay numbers only.
-nc	Do not set constant signals as covered. When -nc is used, they will appear in the uncovered connection list.
-o <file_name[.**twr**]>	The name of the output file to contain the timing report. If no output name is specified, the report name defaults to the same root name as the .ncd file.
-p	Reports the number of asynchronous loops in verbose mode.
-primetime	This will create a primetime format in the report file. This only effects reporting style of path summary and the Constraint Details sections. The Constraint Details information is just organized and presented in a slightly different manner.
-sethld	Allows you to perform a standard setup analysis and a hold time analysis at the same time and generates one report output file containing both setup and hold report. To change default peformance grade for hold time using this option, use the **-sphld** option.
-setuphold	Allows you to perform standard setup analysis and a hold time analysis on I/Os in the same report file. In the summary section, you will see both setup and hold time timing for each of your INPUT_SETUP and CLOCK_TO_OUT preferences.
-sp <peformance>	Overrides the default peformance grade. When not using the **-hld** option to perform hold time checks (i.e., setup time), the default peformance grade will be derived automatically from the .ncd file based upon device selection.
-sphld <peformance>	Overrides the default peformance grade for hold analysis when using the **-sethld** option which performs both setup and hold analysis and generates one report file. Setup analysis appears first, followed by hold analysis.
-v <limit>	The **-v** option generates a verbose (full) timing report file in the current working directory. If this or the **-e** option is unspecified, TRACE sends a summary report to standard output. The <limit> value sets the allowable number of items reported for each timing preference. Valid entries are an integer value that is 0 or greater. If you enter a 0, there will be no limit in the amount of worst case paths reported (0 = no limit).

Examples Following are a few examples of TRACE command lines and a description of what each does.

Example 1 The following command verifies the timing characteristics of the design named design1.ncd, generating a summary timing report. Timing preferences contained in the file group1.prf are the timing constraints for the design. This generates the report file design1.twr.

trce design1.ncd group1.prf

Example 2 The following command produces a file listing all delay characteristics for the design named design1.ncd. Timing preferences contained in the file group1.prf are the timing constraints for the design. The file output.twr is the name of the verbose report file.

trce -v design1.ncd group1.prf -o output.twr

Example 3 The following command analyzes the file design1.ncd and reports on the three worst errors for each preference in timing.prf. The report is called design1.twr.

trce -e 3 design1.ncd timing.prf

Example 4 The following command analyzes the file design1.ncd and produces a verbose report to check on hold times on any FREQUENCY, CLOCK_TO_OUT, INPUT_SETUP and OFFSET preferences in the timing.prf file. With the output report file name unspecified here, a file using the root name of the .ncd file (i.e., design1.twr) will be output by default.

trce -v -hld design1.ncd timing.prf

Example 5 The following command analyzes the file design1.ncd and produces a summary timing report to check on both setup and hold times on any INPUT_SETUP and CLOCK_TO_OUT timing preferences in the timing.prf file. With the output report file name unspecified here, a file using the root name of the .ncd file (i.e., design1.twr) will be output by default.

trce -setuphold design1.ncd timing.prf

Example 6 The following command analyzes the file design.ncd and produces a summary timing report to check on both setup and hold times on any FREQUENCY, INPUT_SETUP, CLOCK_TO_OUT, OFFSET timing preferences in the design.prf file. With the output report file name unspecified here, a file using the root name of the .ncd file (i.e., design.twr) will be output by default. The -sphld option override is used to set the peformance grade on just the hold analysis to minimum.

trce -sp 5 -sethld -sphld M design.ncd design.prf

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Running I/O Timing from the Command Line

The **I/O Timing Report** process in the Diamond environment can also be run through the command line using the **iotiming** program. For each input data port, this process will generate the setup and hold time requirements and min/max clock-to-out delay for each output port. The computation will be performed over all peformance grades and at the voltage and temperature specified in the preference file. This program also automatically determines the clocks and their associated data ports.

Subjects included in this topic:

- Running IOTIMING
- Command Line Syntax
- IOTIMING Options
- Examples

Running IOTIMING IOTIMING uses your input placed-and-routed physical design (.ncd) file and optional associated preference (.prf) file to generate an I/O Timing Report (.ior) file.

- To run IOTIMING, type **iotiming** on the command line with, at minimum, the names of your input .ncd file. You should specify a preference file since factors such as TEMPERATURE, VOLTAGE, and DDR elements affect the I/O timing. Otherwise you may see differences between the generated I/O Timing Report (.ior) and TRACE Timing Report (.twr) files. A sample of a typical IOTIMING command would be as follows:

  iotiming input.ncd

There are several commands options that do not need to be specified due to the default behavior of IOTIMING, so please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **iotiming** [**-s**] [**-v**] [**-sp** <peformance_grade>] <design.ncd> [<preference.prf>] **-o** <output_file>

IOTIMING Options The table below contains descriptions of all valid options for IOTIMING.

Table 908: IOTIMING Command Line Options

Option	Description
<design.ncd>	The input Native Circuit Description (.ncd) file that is used to generate output timing information in the I/O Timing Report .ior file. This design must be placed and routed.
-s	Specifies peformance grade in the NCD file.
-sp <peformance_grade>	Specifies the selected peformance grade.

Table 908: IOTIMING Command Line Options

Option	Description
-o <output_file>	Specifies the name of the output I/O Timing Report (.ior) file that will be generated in the current working directory unless otherwise specified. The default file name of the output if not specified is design.ior. The resulting output file contains sections for Input Setup/Hold Time and also for Clock-to-Out for every available peformance grade for the current device.
<preference.prf>	The input preference (.prf) file that contains any changes in temperature and voltage through the use of the VOLTAGE and TEMPERATURE preferences.
-v	Verbose report option that will produce an I/O Timing Report (.ior) which summarized worst case scenario data on all available peformance grades. If you do not specify this option, the program report defaults to a summary report of the worst case peformance grade for the selected device in addition to any available faster peformance grades that are available for the device. For example, if your device is offered in peformance grades -3, -4, and -5 and your selected device is a -4, then by default your report will include analyses on -4, -5, and M (minimum peformance grade).

Examples Following are a few examples of IOTIMING command lines and a description of what each does.

Example 1 This command takes the input.ncd and input.prf files and generates an io_report.ior report file in the reports directory above the current directory. Here the designer has placed new VOLTAGE and/or TEMPERATE preferences on the design, so specifying the .prf file is necessary. The resulting report will be a summary of the worst case peformance grade.

iotiming input.ncd input.prf -o ../reports/io_report

Example 2 This shortcut command performs the same function on the input.ncd, but it does not use an input preference file and will output an I/O timing report by the default name of input.ior in the current working directory.

iotiming input.ncd

Example 3 This command uses the input.ncd and input.prf files to output a verbose I/O timing report named io_report.ior in the current working directory. A verbose report (**-v**) option generates a report containing data on all available peformance grades.

iotiming -v input.ncd input.prf -o io_report

See Also ▶"Mapping" on page 590

- "Command Line Data Flow" on page 2467
- "Command Line Program Overview" on page 2464

Running Backannotation from the Command Line

The **Generate Timing Simulation Files** process in the Diamond environment can also be run through the command line using the **ldbanno** program. The **ldbanno** program back-annotates physical information (e.g., net delays) to the logical design and then writes out the back-annotated design in the desired netlist format. Input to **ldbanno** is a Native Circuit Description file (.ncd) a mapped and partially or fully placed and/or routed design.

Subjects included in this topic:

- Running LDBANNO
- Command Line Syntax
- LDBANNO Options
- Examples

Running LDBANNO LDBANNO uses your input mapped and at least partially placed-and-routed Native Circuit Description (.ncd) file to produce a back-annotated netlist (.v, .vhd) and standard delay (.sdf) file. This tool supports all FPGA design architecture flows.

- To run LDBANNO, type **ldbanno** on the command line with, at minimum, the type of your output netlist (i.e., Verilog or VHDL) with the **-n** option and the name of your input .ncd file. A sample of a typical LDBANNO command would be as follows:

  ldbanno -n vhdl backanno.ncd

Note

The above command back annotates backanno.ncd and generates a VHDL file backanno.v and an SDF file backanno.sdf. If the target files already exist, they will not be overwritten in this case. You would need to specify the **-w** option to overwrite them.

There are several command line options that give you control over the way LDBANNO generates back-annotated netlists for simulation. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax (VHDL) **ldbanno** [**-w**] [**-pre** <prfx>] [**-sp** <grade>] [**-neg**] [**-pos**] [**-sup**] [**-min**] [**-f** <command_file>] **-h** [<hdltype>] [**-t**] [**-x**] [**-slice**] [**-tpath**] [**-dis** [<del>]] [**-m** [<limit>]] [**-h** vhdl] [**-u**] [**-nopur**] [**-n** <type>] [**-l** <libtype>] [**-noslice**] [**-p** <prffile>] [**-o** <vhdl[**.vhd**]>] [**-d** <delays[**.sdf**]>] [<ncdfile[**.ncd**]>] {<preference[**.prf**]>}

Command Line Syntax (Verilog) **ldbanno** [**-w**] [**-pre** <prfx>] [**-sp** <grade>] [**-neg**] [**-pos**] [**-sup**] [**-min**] [**-f** <command_file>] **-h** [<hdltype>] [**-x**] [**-slice**] [**-t**] [**-dis** [<del>]] [**-m** [<limit>]] [**-u**] [**-i**] [**-nopur**] [**-n** <type>] [**-l** <libtype>] [**-s** <separator>] [**-noslice**] [**-p** <prffile>] [**-o** <verilog<.v>>] [**-d** <delays[**.sdf**]>] [<ncdfile[**.ncd**]>] {<preference[**.prf**]>}

LDBANNO Options The table below contains descriptions of all valid options for LDBANNO.

Table 909: LDBANNO Options

Option	Description
-d <delays[**.sdf**]>	Output delay file, defaults to <ldbfile>.sdf.
[**-dis** [<delay>]] [**-sig** <sig_file>]	The **-dis** option distributes routing delays by splitting the signal and inserting buffers. The <delay> value represents the maximum delay number in picoseconds between each buffer (1000 ps by default) The **-sig** option designates the use of a <sigfile> which is an ASCII file that contains all the top level signals to be split. One signal name can used per line. If **-dis** is given but **-sig** is absent, all top level signals will be distributed. These options will enable simulation of high frequency signals (e.g., clock signals), where the interconnection delay may be higher than the clock cycle. In such a case, that signal will be blocked in some simulators if the delay is not distributed.
-f <command_file>	Execute command line arguments in the specified command_file. See “The –f Option” on page 2583, “Command File Example” on page 2582, and “Using the Command File” on page 2583.
-h [<hdltype>]	The -h help option on the command line can be specified for the values **vhdl** or **verlilog** as some options are limited by the hardware description language being used.
-i	Inserts a buffer at each block input that has interconnection delay on it. **Note**: This option is valid only when the netlist type is Verilog. It is similar to the “tipd” statement in a VITAL compliant model in VHDL, which can help you to debug the design by isolating the interconnection delay and pin-to-pin delay on each input.
-l <libtype>	The orca library is the default for both VHDL and Verilog.
-m <limit>	Shortens the block names to a given character limit in terms of some numerical integer value.
-min	Replaces all timing information for backannotation with the minimum timing for all paths. This option is used for simulation of hold time requirements. Separate simulations are required for hold time verification (**-min** switch) and delay time verification (normal output).

Table 909: LDBANNO Options

Option	Description
-n <type>	Specifies the netlist type for ldbanno to write out : ▶ Verilog generic verilog format with SDF delay file ▶ VHDL generic VHDL format with SDF delay file So, accepted values are **vhdl** and **verilog**. Note that these values are not case sensitive.
-neg	For better compatibility with simulators, all negative setup/hold delay numbers in the SDF file are set to 0 by default. This may cause some discrepancies between backannotation and the TRACE result. Use the new -neg option to get the negative numbers in the SDF file and backannotation will match the TRACE report. Ensure that the simulator is able to handle negative numbers in the SDF file.
-nopur	Option prevents the writing of a PUR instance in the Verilog/VHDL backannotated netlst. You must instantiate it in the test bench.
-noslice	(optional) Verbose netlist option. If this option is excluded, a parameterized netlist format will be generated that is more compact in size and memory efficient within the simulator.
-o <netlist>	(optional) The name of the output file. The extension used for each output depends on the type of netlist being written (specified with the **-n** switch).
-p <prffile>	(optional) Preference file (only output load preferences are used).
-pos	(LatticeSC/M) Writes out zero value for negative setup/hold time.
-pre <prefix>	(optional) Prefix to add to module names to make them unique for multi-chip simulation.
-s <separator>	(<type>=Verilog) Hierarchy separator character.
-slice	(optional) Non-Verbose netlist option. If this option is included, a parameterized netlist format will be generated that is more compact in size and memory efficient within the simulator.
-sp <peformance grade>	Re-target backannotation to a different peformance grade than the one used to create the .ncd file. You are limited to those peformance grades available for the port used in the .ncd file.
-sup	Suppresses custom design MACO block in SDF output.

Table 909: LDBANNO Options

Option	Description
-t <type>	(<type>=VHDL) Target this VHDL file to Exemplar Time Explorer. (<type>=Verilog>) Write a PrimeTime compatible Verilog file.
-tpath	Transport mode of path delay in VHDL.
-u	Adds paths for top-level dangling nets.
-w	Overwrite the output file(s).
-x	If you use the **-x** option, LDBANNO will place "X" notifiers in the output file on flip-flops with setup and/or hold time violations. This option is supported by all new Lattice FPGA families. (<type> = VHDL or Verilog) Write a trce cross-reference file.
-z	Zero delay (do not calculate or write any delays). See Note for the **-a** option.
<preffile[.**prf**]>	(optional) The preference file in .prf format.
<ncdfile[.**ncd**]>	The input design file is a mapped and partially or fully placed and/or routed design in .ncd format.

Examples Following are a few examples of LDBANNO command lines and a description of what each does.

Example 1 The following command back annotates design.ncd and generates a Verilog file design.v and an SDF file design.sdf. If the target files exist, they will be overwritten.

ldbanno -w -n verilog design.ncd

Example 2 The following command back annotates design.ncd and generates a VHDL file backanno.vhd and an SDF file backanno.sdf. Any signal in the design that has an interconnection delay greater than 2000 ps (2 ns) will be split and a series of buffers will be inserted. The maximum interconnection delay between each buffer would be 2000 ps.

```
ldbanno -dis 2000 -n vhdl -o backanno design.ncd
```

Example 3 The following command re-targets backannotation to peformance grade -2, and puts a buffer at each block input to isolate the interconnection delay (ends at that input) and the pin to pin delay (starts from that input).

```
ldbanno -sp 2 -i -n verilog design.ncd
```

Example 4 The following command generates Verilog netlist and SDF files without setting the negative setup/hold delays to 0:

```
ldbanno -neg -n verilog design.ncd
```

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Running LTXT2PTXT from the Command Line (ECP5 and LatticeECP3 Only)

This process converts the logical attribute text file (.txt) for the LatticeECP3 PCS block to the physcial attribute text file (.ptx). The LTXT2PTXT process is run automatically when using the Diamond GUI process flow and does not require user input. But, when when using the command line process flow, this process must be run on the command line before generating a bitstream for ECP5 and LatticeECP3 designs that utilize the PCS block.

Subjects included in this topic:

▶ Running LTXT2PTXT

▶ LTXT2PTXT Command Line Syntax

▶ LTXT2PTXT Options

▶ Example

Running LTXT2PTXT LTXT2PTXT converts the logical attribute text file (.txt) for the ECP5 and LatticeECP3 PCS block to the physcial attribute text file (.ptx).

▶ To run LTXT2PTXT, type **ltxt2ptxt** on the command line.

Command Line Syntax ltxt2ptxt [-path search_path] ncd_file

ltxt2ptxt [-path dir1 -path dir2 ..] ncd_file

LTXT2PTXT Options The table below contains descriptions of all valid options for LTXT2PTXT.

Table 910: LTXT2PTXT Command Line Options

Option	Description
- path	Path where the logical text file resides.

Example Following is an example of running LTXT2PTXT from the command line.

```
ltxt2ptxt -path "c:/lscc/examples" design.ncd
```

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Running Bit Generation from the Command Line

The **Bitstream** process in the Diamond environment can also be run through the command line using the bit generation (**bitgen**) program. This topic provides syntax and option descriptions for usage of the **bitgen** program from the command line. The **bitgen** program takes a fully routed Native Circuit Description (.ncd) file as input and produces a configuration bitstream (bit images) needed for programming the target device.

Subjects included in this topic:

- Running BITGEN
- Command Line Syntax
- BITGEN Options
- Examples

Running BITGEN BITGEN uses your input, fully placed-and-routed Native Circuit Description (.ncd) file to produce bitstream (.bit, .msk, or .rbt) for device configuration.

- To run BITGEN, type **bitgen** on the command line with, at minimum, the **bitgen** command. There is no need to specify the input .ncd file if you run **bitgen** from the directory where it resides and there is no other .ncd present. An output .bit file will be generated if you do not specify a mask file with **-m** or a raw bits file with **-b**. The sample below shows an option added that shows the **-g Address:Explicit** option specified used for testing and debugging of LatticeSC/M devices:

  bitgen -g Address:Explicit

There are several command line options that give you control over the way BITGEN outputs bitstream for device configuration. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **bitgen** [**-b**] [**-d**] [**-jedec**] [**-w**] [**-m** <format>] [**-e**] [**-s** <secfile>] [**-k** <bekfile>] [**-f** <command_file>] [**-h** [<architecture>]] [**-o** <outfile>] [**-x**] [**-path** <search_path>] [**-j**] [**-q**] [**-t**] [**-g** option:value] [**-J r** | **w** <infile1[**.ncd**] {<infile2[**.ncd**]}] <infile[**.ncd**]> [<outfile>] [<preffile[.**prf**]>]

BITGEN Options The table below contains descriptions of all valid options for BITGEN.

Note

Many BITGEN options are only available for certain architectures. Please use the **bitgen** -h <*architecture*> help command to see a list of valid bitgen options for the particular device architecture you are targeting.

Table 911: BITGEN Command Line Options

Option	Description
-b	Create a “rawbits” (.rbt) file in ASCII format instead of a binary file. Do not use **-b** with **-m** if you want both a rawbits file and a mask file (see the **-m** command below). Instead run BITGEN twice once with the **-b** option and once with the **-m** option.
-d	Do not run DRC.
-e	(LatticeECP2S, LatticeXP2) Create an encrypted bitstream file. This must be used with the **-s** and **-k** options that specify binary security and key files. These binary files are generated using the Security Setting tool, **secuload**. Here is the basic syntax for using the **-e** option, bitgen -e -s <sec_file> -k <bek_file> See related topics under the “Setting Security Options” on page 76 topic for details on using **secuload** to generate the associated binary security (.sec) and key (.bek) files. The necessary binary key and security files for encryption can also be generated in the Diamond GUI using the **Tools > Security Setting** menu option.
-ebrinit <format>	(LatticeECP2 Write out Embedded Block RAM (EBR) initial values. Options are **0** which prevents bitgen from writing any initial EBR values, **1** (default) which writes out initial values for those used EBRs with valid WID, and **2**, which writes out initial values for all EBRs.
-f <command_file>	Execute command line arguments in the specified command_file. See “The –f Option” on page 2583, “Command File Example” on page 2582, and “Using the Command File” on page 2583.
-g <option>:<value>	The -g option sets FPGA configuration options. Options and values are case-insensitive. *Valid -g option values are described in the following rows.*
-g Address:<value>	ADDRESS (LatticeSC/M) Bitstream addressing mode. Options are **Increment** (default) for general use and **Explicit** for testing and debugging. bitstream has two Address modes: The **Increment** mode contains data information but no address information and each frame is loaded from lowest address to highest address. The **Explicit** mode requires data information followed by address information, one address frame or more (depending upon how many locations are to be written). The default option for Address is **Increment**. There is also an option in Explicit mode to ignore the zero frames. See the ZEROFRAMES option.

Table 911: BITGEN Command Line Options

Option	Description
-g CfgMode:<value>	CFGMODE (LatticeSC/M) Options are **Enable** (default) and **Serial**. If set to Serial, the configuration mode will be slave serial during reconfiguration, regardless of the setting of the MODE pins. (LatticeECP/EC, LatticeXP) Options are **Disable** (default), **Flowthrough**, and **Bypass**. The Flowthrough and Bypass options are for device chains. Each of these options places an instruction at the end of the bitstream. During configuration, such an instruction tells the device not to wake up because there is more data coming. If set to Flowthrough, the instruction tells the device that the data should be ignored because it is for the next device in the chain. If set to Bypass, it tells the device to read the data and serially shift it out to the next device.
-g DisableUES:<value>	(LatticeXP2) Options are **False** (default) and **True**.
-g DoneActive:<value>	DONEACTIVE Select the event that activates the DONE signal. Settings are **C1**, **C2**, **C3**, **C4** (default) (first CCLK rising edge after the length count is met, second CCLK rising edge after the length count is met, etc.).
-g DonePhase:<value>	DONEPHASE (ECP5, LatticeECP/EC, LatticeECP2/M/S, LatticeXP, LatticeXP2, LatticeECP3) This option determines the period of the wake-up sequence in which the Done pin is released or actively driven. Optional values are **T3** (default), **T2**, **T1**, and **T0**. T0 is the first sequence in the wakeup sequence, followed by T1, T2, and T3.
-g Es:<value>	ES (ECP5, LatticeECP/EC, LatticeECP2/M/S, LatticeXP, LatticeXP2, LatticeSC/M, LatticeECP3) Engineering Sample (ES) part option. Specify this option if the part you are generating bitstream for is an ES version of the chip opposed to the final production version. Optional values are **No** (default) and **Yes**.
-g ExternalClk:<value>	EXTERNALCLK Optional values are **No** (default) and **Yes**.
-g GoEphase:<value>	GOEPHASE (LatticeECP/EC, LatticeECP2/M/S, LatticeXP, LatticeXP2) (LatticeSC/M Only) This option determines the period of the wake-up sequence in which the Global Output Enable becomes active. Optional values are **T3**, **T2**, and **T1** (default).

Table 911: BITGEN Command Line Options

Option	Description
-g GrantTimeout:<value>	GRANTTIMEOUT (LatticeSC/M) Controls an internal system bus grant counter which counts the number of HCLK (sysbus clk) cycles goes by before the grant signal is taken away from the master.This prevents the system bus from locking up if anything goes wrong. Possible values are **0, 1, 2...15**. The default value is 5. See WAITSTAITTIMEOUT.
-g GsrPhase:<value>	GSRPHASE (ECP5, LatticeECP/EC, LatticeECP2/M/S, LatticeXP, LatticeXP2, LatticeECP3) This option determines the period of the wake-up sequence in which the Global Set/Reset becomes active. Optional values are **T2** (default), **T3**, and **T1**.
-g GwdPhase:<value>	GWDPHASE (ECP5, LatticeECP/EC, LatticeECP2/M/S, LatticeXP, LatticeXP2, LatticeECP3) This option determines the period of the wake-up sequence in which the Global Write Disable becomes active. Optional values are **T2** (default), **T3**, and **T1**.
-g GsrInactive:<value>	GSRINACTIVE Select the event that releases the internal set/reset to the latches and flip-flops. Optional values are **C1**, **C2** (default), **C3**, **C4**, and **Donein**.
-g Jtag:<value>	JTAG Enables or disables JTAG (boundary scan) after configuration. Optional values are **Enable** and **Disable** (default).
-g LengthBits:<value>	LENGTHBITS (LatticeSC/M) Specifies the number of bits in the bitstream length field. Options are **24** (default) and **32**.
-g Oscillator:<value>	OSCILLATOR Enables the on-chip oscillator after configuration. Options are **Enable**, **Disable** (default), and **EnableDiv8** (divide the speed by eight).
-g OutputsActive:<value>	OUTPUTSACTIVE Select the event that releases the I/O from 3-state condition and turns the configuration-related pins operational. Settings are **C1**, **C2**, **C3**, **C4**, and **Donein** (when the DONEIN signal goes high the default).
-g OutputsCfg:<value>	OUTPUTSCFG (LatticeSC/M) Optional values are **TriStateDuringReConfig** (default) and **DriveDuringReConfig**. If **TriStateDuringReConfig** is specified**,** the FPGA will not drive outputs during configuration/reconfiguration.

Table 911: BITGEN Command Line Options

Option	Description
-g RAMCfg:<value>	RAMCFG (LatticeSC/M) Optional values are **ClearDuringReConfig** (default). **ClearDuringReConfig** reinitializes the device when you download bitstream. **NoClearDuringReConfig** retains the current configuration and allows additional bitstream configuration. (ECP5, MachXO, LatticeECP/EC, LatticeECP2/M/S, LatticeXP, LatticeXP2, LatticeECP3) Optional values are **Reset** (default) and **NoReset**. **ClearDuringReConfig** reinitializes.
-g ReadBack:<value>	READBACK Allows you to extract the configuration data stored in an FPGA in order to verify the configuration. Settings are **Disable** (default), **Once** (allow one readback only; after that, readback cannot be invoked again), and **Command** (multiple readbacks). (LatticeXP Only) Optional values are **FLASH** (default) and **SRAM**.
-g ReadCapture:<value>	READCAPTURE Enable or disable readback of the configuration bitstream. Settings are **Disable** (default), **Read**, **UserNet**, and **Both**. (ECP5, LatticeECP/EC and LatticeXP Only) Optional values are **Disable** (default) and **Enable**. When Enable is selected, the GLB is targeted during readback.
-g RegisterCfg:<value>	REGISTERCFG Reset the PFU registers before configuration. Values are **Enable** (default) and **Disable**. (For LatticeSC Only) Optional values are **ResetDuringReConfig** (default) and **NoResetDuringReConfig**.
-g Rnet: <value>	RNET (LatticeSC/M) INIT, ON, OFF.
-g SPI_RECONFIG_START_ADDRESS:<value>	SPI_RECONFIG_START_ADDRESS (LatticeSC/M) Address of the next programming bitstream to be loaded from SPI Flash boot memory. It is 6 or 8 digit hex string prefixed with 0x. The default address value is 0x00000000.
-g StartupClk:<value>	STARTUPCLK (LatticeSC/M) Use CCLK or a User clock during startup. Options are **Cclk** (default) or **UserClk**.
-g SyncToDone:<value>	SYNCTODONE Synchronize the startup sequence to the external DONE signal. Optional values are **No** (default) and **Yes**.

Table 911: BITGEN Command Line Options

Option	Description
-g Sysbusconfig:<value> **-g SysbusCFG**:<value>	SYSBUSCFG (Lattice SC/M) Optional values are **ResetDuringReConfig** (default), **NoResetDuringReConfig**. The system bus is either reset or not reset at reconfiguration.
-g SysbusClkconfig:<value>	SYSBUSCLKCFG (Lattice SC/M) Optional values are **ResetDuringReConfig** (default), **NoResetDuringReConfig**. The system bus clock is either reset or not reset at reconfiguration.
-g WaitStateTimeout:<value>	WAITSTATETIMEOUT (LatticeSC/M) Controls an internal system bus timeout counter. This counter counts the number of HCLK (sysbus clk) cycles goes by while the system bus is in wait states. Possible values are **0, 1, 2...15**. 0 - Indefinitely (never times out) 1 - 2^2 HCLK cycles 2 - 2^4 HCLK cycles 3 - 2^6 HCLK cycles 4 - 2^8 HCLK cycles 5 - (default) 2^10 HCLK cycles 6 - 2^12 HCLK cycles 7 - 2^14 HCLK cycles 8 - 2^16 HCLK cycles 9 - 2^18 HCLK cycles 10- 2^20 HCLK cycles 11 - 2^22 HCLK cycles 12 - 2^24 HCLK cycles 13 - 2^26 HCLK cycles 14 - 2^28 HCLK cycles 15 - 2^31 HCLK cycles

Table 911: BITGEN Command Line Options

Option	Description
-g Xres:<value>	XRES (LatticeSC/M) Controls the calibration circuit connected to XRES pin. Options are **INIT** (default), **ON**, and **OFF**. The external resistor combined with an internal calibration control circuit is used to calibrate output drive strength and impedance. The calibration compensates for the effects of variations in process, voltage and temperature (PVT). It is connected through a 1K+/-1% resistor to ground. The **INIT** options allows the calibration circuit to monitor PVT on power-up and during configuration. The circuit latches in the control value and not allow changes after that. If voltage and temperature does not vary much during operation versus configuration time, INIT is the best choice. The **ON** option allows the calibration circuit to run all the time. If IOPVTCTRL is instantiated in the design, the XRES is controlled by the UPDATE signal. INIT and OFF options are disabled. The **OFF** option turns off the calibration circuit. The result is minimum drive on output buffer, maximum impedance and maximum termination resistor. OFF is not recommended in most FPGA implementations.
-g ZeroFrames:<value>	ZEROFRAMES (LatticeSC/M) Option to specify that data frames with all zeros be written into the bitstream. Options are **No** and **Yes** (default). This option is used with Increment Address mode. When **Yes** is specified with Increment Address mode, every data frame is written out with no address frames. When **No** is specified with Increment Address mode, all sequential non-zero data frames are written without data frames. When a (zero) data frame is skipped, the address frame for the next (non-zero) data frame is written.
-h <architecture> or **-help** <architecture>	Display available BITGEN command options for the specified architecture. The **bitgen -h** command with no architecture specified will display a list of valid architectures.
<infile[.**ncd**]>	A .mif file will be generated with a file name identical to your module name. Note that if block RAMs are already initialized by INITVAL properties in the .ncd file, addresses that are explicitly addressed in the .mif file will be overridden by the INITVAL properties.
-j	Do not create a bitstream file.
-jedec	(LatticeXP2) Creates a JEDEC format file.

Table 911: BITGEN Command Line Options

Option	Description
-J r | **w** <infile1[**.ncd**]> {<infile2[**.ncd**]}	JTAG setup. Allows the user to set JTAG port read and write on the FPGA chip by providing them with the setup bitstreams. Users will use tools other than ispVM to download these bitstreams. Example syntax: [**-J w** <infile1.ncd> {<infile2.ncd>}] [**-J r**] [**-a**] [**-o** <outfile>] where: **-J r** generates a JTAG read setup bitstream. The default output is jtagread.jbt. **-J w** generates JTAG write setup bitstream. **-a** outputs an ASCII file. **-o** <outfile> specifies the output file name. The example syntax for JTAG setup will generate JTAG write setup bitstream that specifies infile1.ncd as the target design to be downloaded, so the correct initialization bits will be added to the bitstream. If more than one design is on the command line, the bitstreams for daisy-chained devices will be generated using JTAG port. The default output is infile1.jbt.
-k <bek_file>	(LatticeECP2S, LatticeXP2) Only when using the **-e** encryption option, use the **-k** option to specify a binary key file. This .bek file is generated by the Security Setting tool, **secuload**. You must also specify the **-s** <sec_file> option in this command to include the related security file. For example, bitgen -e -s <sec_file> -k <bek_file> See related topics under the "Setting Security Options" on page 76 for details on using **secuload** to generate the binary key and security files. The necessary binary key and security files for encryption can also be generated in the Diamond GUI using the **Tools > Security Setting** menu option.

Table 911: BITGEN Command Line Options

Option	Description
-m <format>	Create mask files and readback files. A mask file is an ASCII file with only data frames in it. When you specify the **-m** option, **bitgen** generates a mask file for the input design. All **bitgen** options can be used with mask file option with the exception that any compressed options are ignored by mask file generation. (For LatticeSC/M, MachXO) Format value equals **0**, **1**, **2**, or **3**. The **0** value outputs files in ASCII. The value **1** outputs files in binary. The value **2** creates mask out SMI MEM cells in some special tiles and outputs files in ASCII. The value **3** creates mask out SMI MEM cells in some special tiles and outputs files in binary. There are three different mask file outputs. They are all similar in format with slight differences in content. The mask file has the following format: 01[data frame][alignment bits][checksum]11111111 (For ECP5, LatticeECP/EC, LatticeXP, LatticeXP2, LatticeECP3) Format value equals **0** or **1**. The **0** value outputs files in ASCII. The **1** value outputs files in binary. **Generic format** – In the date frames, the Vs will be positioned at the location of every configuration RAM location that is used for the device family. The Xs are positioned at the location of ever configuration RAM location that is unused for the device family. The alignment bits are 0's and the checksum is affected by relative location of the mask bits in a byte of the frame. If a mask bit is at bit 2 of byte 3 for the frame, bit 2 at the checksum is masked as “X” as well. **Design specific format** – The design specific mask file is an ASCII file containing 1s and 0s, and Xs. This file has the same structure as the Generic mask file except instead of a V in all used configuration RAM locations the actual programming data is in the appropriate locations in the data frame. The Xs are at the locations of the unused configuration RAM as in the Generic file. **Design specific format with masks for user RAM** – This format is also an ASCII file that follows the format of the design specific file with the addition of Xs in the locations of any PFUs configured as user ram in the design. This is any RAM blocks that the user has put in his design.
-o <outfile>	Specifies the output file name.

Table 911: BITGEN Command Line Options

Option	Description
<outfile>	The output file. If you do not specify an output file, BITGEN creates one in the input file's directory. If you specify **-m** on the command line, the extension is .msk. If you specify **-b**, the extension is .rbt. Otherwise the extension is .bit. A report (.bgn) file containing all of BITGEN's output is automatically created under the same directory as the output file.
-path <search_path>	(LatticeSC/M) Use to specify a search path to the directory in which PCS configuration files are located. To specify multiple **-path** options, precede each with **-path**; or combine multiple search_path specifiers after one **-path**. The syntax for combining multiple search_path specifiers for a given macro search path is as follows where dir is the directory search path: **-path** dir1;dir2;dir3 (for Windows) **-path** dir1:dir2:dir3 (for UNIX) For example, you may have to specify a path for some modules you generate in IPexpress (e.g., PCS and some Sysbus for LatticeSC), because **bitgen** needs their configuration option information to output your bitstream. Since IPexpress allows you to specify a directory other than your project directory to keep output files in, **bitgen** requires this option to find this information. See the IPexpress Help for details.
-q	Creates .bit output without header comments (for special use only).
-s <sec_file>	(LatticeECP2S, LatticeXP2) Only when using the **-e** encryption option, use the **-s** option to specify a binary security file. This .sec file is generated by the Security Setting tool, **secuload**. You must also specify the **-k** <bek_file> option in this command to include the related key file. For example, bitgen -e -s <sec_file> -k <bek_file> See related topics under the "Setting Security Options" on page 76 for details on using **secuload** to generate the binary security and key files. The necessary binary security and key files for encryption can also be generated in the Diamond GUI using the **Tools > Security Setting** menu option.
-t	(ECP5, LatticeECP/EC, LatticeXP, LatticeECP2/M, LatticeXP2, LatticeECP3) Tie down. This option ties down unused routing MUXes to a ground state, avoiding unnecessary toggling and reducing power consumption.
-w	Overwrite an existing .bit, .msk, or .rbt output file.
-x	Byte mirror. Swaps 0s with 7s, etc.

Examples Following are a few examples of BITGEN command lines and a description of what each does.

Example 1 The following command tells **bitgen** to overwrite any existing bitstream files with the **-w** option, prevents a physical design rule check (DRC) from running with **-d**, specifies a raw bits (.rbt) file output with **-b**, and specifies a **-g** option for PERSIST to be turned on. Notice how these three options can be combined with the **-wdb** syntax.

bitgen -wdb -g PERSIST:Yes

Example 2 The following command specifies a raw bits (.rbt) file output with **-b**, specifies several **-g** options for disabling features, and specifies a certain .ncd file located in the present working directory because there are several .ncd files present. The name of the output .rbt file is also specified in this case.

bitgen -b -g JTAG:Disable ReadBack:Disable ReadCapture:Disable routed.ncd output_bit.rbt

Example 3 The following command shows bitgen calling a command file named bit1d. This file can contain any number of bitgen commands. See The -f Option.

bitgen -f bit1d

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Running Programmer from the Command Line

You can run Programmer from the command line. The **PGRCMD** command uses a keyword preceded by a hyphen for each command line option.

Subjects included in this topic:

- Running PGRCMD
- Command Line Syntax
- PGRCMD Options
- Examples

Running PGRCMD PGRCMD allows you to download data files to an FPGA device.

- To run PGRCMD, type **pgrcmd** on the command line with, at minimum, the **pgrcmd** command.

There are several command line options that give you control over the way PGRCMD programs devices. Please refer to the rest of the subjects in this topic for more details.

Command Line Syntax The following describes the PGRCMD command line syntax:

pgrcmd [**-help**] [**-infile** <input_file_path>] [**-logfile** <log_file_path>] [**-processtype** <turbo|sequential> [**-cabletype** <cable>]

-cabletype

lattice [**-portaddress** < 0x0378 | 0x0278 | 0x03bc | 0x<*custom address*> >]

usb [**-portaddress** < EZUSB-0 | EZUSB-1 | ... | EZUSB-15 >]

usb2 [**-portaddress** < FTUSB-0 | FTUSB-1 | ... | FTUSB-15 >]

TCK [**0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10**]

PGRCMD Options The following are PGRCMD options.

Help (Optional)

Option	Description
-help or **-h**	Displays the Programmer command line options.

Input File (required)

Option	Description
-infile *filename*.xcf	Specifies the chain configuration file (.xcf). If the file path includes spaces, enclose the path in quotes.

Log File (optional)

Option	Description
-logfile *logfilename*.log	Specifies the location of the Programmer log file.

Process Type (optional)

Option	Description
-processtype sequential	Processes using Sequential mode (default).
-processtype turbo	Processes using Turbo mode.

This option is only valid when you process directly to the cable. It is not valid with the -outfiletype option.

Cable Type (optional)

Option	Description
-cabletype lattice	Lattice HW-DLN-3C parallel port programming cable (default).
-cabletype usb	Lattice HW-USBN-2A USB port programming cable.
-cabletype usb2	Lattice FHW-USBN-2B (FTDI) USB programming cable and any FTDI based demo boards.

Parallel Port Address (optional)

Option	Description
-portaddress 0x0378	LPT1 parallel port (default)
-portaddress 0x0278	LPT2 parallel port
-portaddress 0x03BC	LPT3 parallel port
-portaddress 0x<*custom address*>	Custom parallel port address

This option is only valid with parallel port cables.

USB Port Address (optional)

Option	Description
-portaddress EZUSB-0 ... EZUSB-15	HW-USBN-2A USB cable number 0 through 15
-portaddress FTUSB-0 ... FTUSB-15	FTDI based demo board or FTDI USB2 cable number 0 through 15

Default is EZUSB-0 and FTUSB-0. Only valid with the USB port cables.

FTDI Based Demo Board or Cable Frequency Control (optional)

Option	Description
-TCK 0, 1, 2, 3, 4, ,5 ,6 7, 8, 9, 10	0 = 30 Mhz 1 = 15 Mhz (default) 2 = 10 Mhz 3 = 7.5 Mhz 4 = 6 Mhz 5 = 5 Mhz 6 = 4 Mhz 7 = 3 Mhz 8 = 2 Mhz 9 = 1 Mhz 10 = 900 Khz

Only applicable for FTDI based demo boards or programming cable.

Return Codes

Code	Definition
0	Success
-1	Log file error
-2	Check configuration setup error
-3	Out of memory error
-4	NT driver error
-5	Cable not detected error
-6	Power detection error
-7	Device not valid error
-8	File not found error

Code	Definition
-9	File not valid error
-10	Output file error
-11	Verification error
-12	Unsupported operation error
-13	File name error
-14	File read error
-17	Build SVF file error
-18	Build VME file error
-19	Command line syntax error

Examples The following is a PGRCMD example.

```
pgrcmd -infile c:\test.xcf
```

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Running the Deployment Tool from the Command Line

You can run the Deployment Tool from the command line. The **DDTCMD** command uses a keyword preceded by a hyphen for each command line option.

Note

If you use the Deployment Tool GUI, the correct command line for the operation your are performing will be displayed in the GUI in the "Step 4 of 4 - Generate Deployment" dialog box. The command line is also recorded in the Deployment Project File (*.ddt). You can view the Deployment Project File with a text editor and view or copy the command line.

Subjects included in this topic:

▶ Running DDTCMD

▶ Command Line Syntax

▶ DDTCMD Options

▶ Examples

Running DDTCMD DDTCMD allows you to convert data files to other formats for one device at a time and devices in a chain. DDTCMD can also use the data files it produces to generate other data file formats.

- To run DDTCMD, type **ddtcmd** on the command line with, at minimum, the **ddtcmd** command.

There are several command line options that give you control over the way **DDTCMD** converts data files. Please refer to the rest of the subjects in this topic for more details.

Command Line Syntax The following describes the DDTCMD command line syntax:

<*install_path*>**ddtcmd**

[**-h**] |

-oft <**-bit | rbt** >

[**-dev** <Device Name>]

-if "<Data File Path>"

[-**of** "<Data File Path>"]

[**-compress** <on | off>]

[**-verifyid** <on | off>]

[**-frequency**]

[**-crc <**frame | global>]

[**-mirror**]

[**-header**]

[**-secure** <on | off>]

[**-overwriteues on** <Hex Value>]

[**-encryption -key hex** <Valid Key Value> | **-keyfile** <Valid Key File>**-ekeypass** <file password>

[**-authentication -pub** <Valid Public Key File> **-prv** <Valid Private Key File> **- eprvpass** <file password> | **-sig** <Valid Signature Key File> **- eprvpass** <file password>

-config_mode <slave_scm | slave_pcm | jtag | spi | spim | master_pcm | slave_spi>]

|

-oft <**-int | mot | xtek**>

[**-dev** <Device Name>]

-if "<Data File Path>"

[-**of** "<Data File Path>"]

[**-compress** <on | off>]

[**-verifyid** <on | off>]

[**-frequency**]

[**-crc <**frame | global>]

[**-mirror**]

[**-header**]

[**-secure** <on | off>]

[**-overwriteues on** <Hex Value>]

[**-encryption -key hex** <Valid Key Value>

-config_mode <slave_scm | slave_pcm | jtag | spi | spim | master_pcm | slave_spi>]

[**-address** <hex address>]

[**-fast**]

|

-oft -svf

[**-dev** <Device Name>]

-if "<Data File Path>"

[**-of** "<Data File Path>"]

-op <Operation>

[**-nocomments**]

[**-revd**]

[**-runtest**]

[**-skipverify**]

[**-reset**]

[**-maxdata** <8 | 16 | 32 | 64 | 128 | 256>]

|

-oft -svfchain

-if "<Data File Path>"

[**-of** "<Data File Path>"]

[**-nocomments**]

[**-revd**]

[**-runtest**]

[**-skipverify**]

[**-reset**]

[**-maxdata** <8 | 16 | 32 | 64 | 128 | 256>]

|

-oft -stp

[**-dev** <Device Name>]

-if "<Data File Path>"

[**-of** "<Data File Path>"]

-op <Operation>

[**-nocompress**]

[**-print**]

[**-skipverify**]

|

-oft -stpchain

-if "<Data File Path>"

[**-of** "<Data File Path>"]

[**-nocompress**]

[**-print**]

[**-skipverify**]

|

oft -ate

-if "<Data File Path>"

[**-of** "<Data File Path>"]

-type <tst | gr | hp3070 | hp3065 | t1800 | t200>

[**-ispdcd**]

[**-skipverify**]

[**-headerfile** <file path>]

[**-splitfile** [**-init**] [**-vectors** <# vectors/file>] [**-splitatlow**]]

|

-oft -fullvme

-if "<Data File Path>"

[**-of** "<Data File Path>"]

[**-maxdata** <8 | 16 | 32 | 64 | 128 | 256>]

[**-hex**]

[**-nocompress**]

[**-compact**]

[**-fixedpulse**]

[**-verifyues**]

[**-noheader**]

[**-comment**]

|

-oft -slimvme

-if "<Data File Path>"

[**-ofa** <Algorithm File Path>]

[**-ofd** "<Data File Path>"]

[**-nocompress**]
[**-hex**]
|
-oft -sspi
-if "<Data File Path>"
[**-ofa** <Algorithm File Path>]
[**-ofd** "<Data File Path>"]
[**-nocompress**]
[**-hex**]
[**-op** <Operation>]
|
-oft -i2c
-if "<Data File Path>"
[**-ofa** <Algorithm File Path>]
[**-ofd** "<Data File Path>"]
[**-nocompress**]
[**-hex**]
[**-op** <Operation>]
-address <80 | (User Specified*)*>
[**-comment**]
[**-fixedpulse**]
|
-oft -cpu
-if "<Data File Path>"
[**-of** "<Data File Path>"]
- type <cpu | c | hex | txt>
[**-verify**]
[**-comment**]
[**-nocompress**]
[**-mirror**]
|
-oft -boot
[**-dev** <Device Name>]
-golden "<Data File Path>"
-primary "<Data File Path>"
-format <int | mot | xtek>

[**-of** "<*D*ata File Path>"]

-flashsize <1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256>

[**-protect**]

[**-mirror**]

[**-header**]

[**-optimize**] [**-fast** | **-dualio** | **-quad 1**]

[**-asc** <1 | 2 | 3 | 4 | 5 | 6 | 7 | 8>

-ascfile “<Data File Path>”

[**-ascfile** “<Data File Path>”] … [**-ascfile** “<Data File Path>”]]

|

-oft -jed

-if “<Data File Path>”

[**-dev** <Device Name>]

[**-of** “<Data File Path>”]

[**-overwriteues** <**on** "<hex value>" | **checksum** | **disable**>]

[**-comment** “<Header File Path>”]

[**-encryption -key hex** “<valid key value>”]

|

-oft -advanced

[**-dev** <Device Name>]

-if “<Data File Path>”

-format <int | mot | xtek>

[**-of** “<Data File Path>”]

-flashsize <512 | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 513>

[**-mirror**]

[**-header**]

[**-optimize**]

[**-userdata** <1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16>

-usermirror

-userfile “<Data File Path>”

-address <hex address value>

[**-userfile** “<Data File Path>”

-address <hex address value>] ...

[**-userfile** “<Data File Path>”

-address <hex address value>]]

[**-ice** [**-warm** <1 | 2 | 3 | 4>]
-patterns <2,3,4>
-bootfile “<Data File Path>”
-address <hex address value>
[**-bootfile** “<Data File Path>”
-address <hex address value>] ...
[**-bootfile** “<Data File Path>”
-address <*hex address value*>]]
[**-fast** | **-dualio | -quad 1**>]
[**-multi** <1 | 2 | 3 | 4>
-golden "<Data File Path>"
-altfile “<Data File Path>”
-address <hex address value>
-next <primary | alt1 | alt2 | alt3 | alt4> [
-altfile “<Data File Path>”
-address <hex address value>
-next <primary | alt1 | alt2 | alt3 | alt4>] ...
[**-altfile** “<Data File Path>”
-address <hex address value>
-next <primary|alt1|alt2|alt3|alt4>]]

|

-oft -merge
-ifdev1 “<Data File Path>”
-ifdev2 “<Data File Path>”
-format <int | mot| xtek>
[**-of** “<Data File Path>”]
[**-mergeformat** <intelligent | combine>
[**-frequency** <freq>]
[**-scoutput** <dout | qout>]]
[**-mirror**]
[-**header**]

|

-oft -bsm
-ifd “<Data File Path>”
-ifb “<BSDL File Path>”
[**-dev** <Device Name>]

[**-of** “<Data File Path>”]

[**-convertbidi**]

DDTCMD Options The following are DDTCMD options.

Option	Description
-h	Displays the Deployment Tool command line usage in the terminal or DOS window.
-oft <File Type>	Indicates which file type(s) will be generated.

Option	Description
-bit	Generate binary bitstream file. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LFE3-17EA-XXFN484` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default file name is the same name as the input file. binary bitstream files should end with the *.bit extension. **-compress** <on | off>: Specifies compression (optional). Default is off. **-verifyid**: On: Include the Verify ID frame. Off: Replaces Verify ID frame with NOOP (optional). Default is off. **-frequency**: Specifies the frequency of the master bitstream. Possible frequencies valid for LatticeEC/P, LatticeECP2/M, LatticeECP3, and LatticeXP/2 devices only are: 2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60, and 130 MHz. Possible frequencies valid for for MachXO2, MachXO3D, and MachXO3L devices only are: 2.1, 2.5, 3.2, 4.3, 5.5, 7, 8.3, 9.2, 10, 13, 15, 20, 27, 30, 33, 38, 44, 53, 66, 89, and 133 MHz. Possible frequencies valid for for ECP5 devices only are: 2.4, 3.2, 4.1, 4.8, 6.5, 8.2, 9.7, 12.9, 15.5, 16.3, 19.4, 20.7, 25.8, 31, 34.4, 38.8, 44.3, 51.7, 62, 77.5, 103.3, and 155 MHz. **-crc** <frame | global>: The "frame" option includes the CRC checking for each data frame. The "global option disables the frames CRC but still calculates the global CRC at the end of the configuration data. Default uses the settings of the input file. Valid for ECP5. LatticeECP2/M, LatticeECP2S/SM, LatticeECP3, MachXO3D, MachXO3L, MachXO2, and LatticeXP/2 devices only. **-secure** <on | off>: Specifies program security (optional). On: Programs the Security Fuse (blocks read back). Off: Does not program the Security Fuse (allows read back).Default uses the settings of the input file. **-overwriteues on**: Overwrites (or replaces) the USERCODE value contained in the input file with the hex value passed in. **Note**: The encryption software patch must be installed before using the following **-encryption -key** and **-config_mode** commands. **-encryption -key hex** <Valid Key Value>: Encrypts the input file using the -key 128-bit encryption key. **-config_mode** <slave_scm | slave_pcm | jtag | spi | spim | master_pcm | slave_spi>]: Selects the appropriate configuration mode for the bitstream encryption. Valid for LatticeECP2S/MS and LatticeECP3 devices only.

Option	Description
-rbt	Generate ASCII Bitstream file. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LFE3-17EA-XXFN484` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default file name is the same name as the input file. ASCII bit files should end with the *.rbt extension. **-compress** <on | off>: Specifies compression (optional). Default is off. **-verifyid**: On: Include the Verify ID frame. Off: Replaces Verify ID frame with NOOP (optional). Default is off. **-frequency**: Specifies the frequency of the master bitstream. Possible frequencies valid for LatticeEC/P, LatticeECP2/M, LatticeECP3, and LatticeXP/2 devices only are: 2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60, and 130 MHz. Possible frequencies valid for for MachXO2, MachXO3D, and MachXO3L devices only are: 2.1, 2.5, 3.2, 4.3, 5.5, 7, 8.3, 9.2, 10, 13, 15, 20, 27, 30, 33, 38, 44, 53, 66, 89, and 133 MHz. Possible frequencies valid for for ECP5 devices only are: 2.4, 3.2, 4.1, 4.8, 6.5, 8.2, 9.7, 12.9, 15.5, 16.3, 19.4, 20.7, 25.8, 31, 34.4, 38.8, 44.3, 51.7, 62, 77.5, 103.3, and 155 MHz. **-crc** <frame | global>: The "frame" option includes the CRC checking for each data frame. The "global option disables the frames CRC but still calculates the global CRC at the end of the configuration data. Default uses the settings of the input file. Valid for LatticeECP2/M, LatticeECP2S/SM, LatticeECP3, ECP5, MachXO2, MachXO3D, MachXO3L, and LatticeXP/2 devices only. **-secure** <on | off>: Specifies program security (optional). On: Programs the Security Fuse (blocks read back). Off: Does not program the Security Fuse (allows read back).Default uses the settings of the input file. **-overwriteues on**: Overwrites (or replaces) the USERCODE value contained in the input file with the hex value passed in. **Note**: The encryption software patch must be installed before using the following **-encryption -key** and **-config_mode** commands. **-encryption -key hex** <Valid Key Value>: Encrypts the input file using the -key 128-bit encryption key. **-config_mode** <slave_scm | slave_pcm | jtag | spi | spim | master_pcm | slave_spi>]: Selects the appropriate configuration mode for the bitstream encryption. Valid for LatticeECP2S/MS and LatticeECP3 devices only.

Option	Description
-int	Generate Intel 32-bit hex file. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LFE3-17EA-XXFN484` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default file name is the same name as the input file. Intel files should end with the *.mcs extension. **-compress** <on | off>: Specifies compression (optional). Default is off. **-verifyid**: On: Include the Verify ID frame. Off: Replaces Verify ID frame with NOOP (optional). Default is off. **-frequency**: Specifies the frequency of the master bitstream. Possible frequencies valid for LatticeEC/P, LatticeECP2/M, LatticeECP3, and LatticeXP/2 devices only are: 2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60, and 130 MHz. Possible frequencies valid for for MachXO2, MachXO3D, and MachXO3L devices only are: 2.1, 2.5, 3.2, 4.3, 5.5, 7, 8.3, 9.2, 10, 13, 15, 20, 27, 30, 33, 38, 44, 53, 66, 89, and 133 MHz. Possible frequencies valid for for ECP5 devices only are: 2.4, 3.2, 4.1, 4.8, 6.5, 8.2, 9.7, 12.9, 15.5, 16.3, 19.4, 20.7, 25.8, 31, 34.4, 38.8, 44.3, 51.7, 62, 77.5, 103.3, and 155 MHz. **-crc** <frame | global>: The "frame" option includes the CRC checking for each data frame. The "global option disables the frames CRC but still calculates the global CRC at the end of the configuration data. Default uses the settings of the input file. Valid for ECP5, LatticeECP2/M, LatticeECP2S/SM, LatticeECP3, MachXO3D, MachXO3L, MachXO2, and LatticeXP/2 devices only. **-mirror**: Flips each byte. Default: bytes are not flipped. Optional. **-header**: Retains the input file header in the output file. Default: Replaces header with all ones (0xFF). Optional. **-secure** <on | off>: Specifies program security (optional). On: Programs the Security Fuse (blocks read back). Off: Does not program the Security Fuse (allows read back).Default uses the settings of the input file. **-overwriteues on**: Overwrites (or replaces) the USERCODE value contained in the input file with the hex value passed in. **Note**: The encryption software patch must be installed before using the following **-encryption -key** and **-config_mode** commands. **-encryption -key hex** <Valid Key Value>: Encrypts the input file using the -key 128-bit encryption key.

Option	Description
-int (cont.)	**-config_mode** <slave_scm | slave_pcm | jtag | spi | spim | master_pcm | slave_spi>]: Selects the appropriate configuration mode for the bitstream encryption. Valid for LatticeECP2S/MS and LatticeECP3 devices only. **-address** <hex address> Specifies the starting address of the memory device to store the input file. **-fast**: Using this option, the FPGA will issue the Fast Read command (0x0B) to the SPI Flash, instead of the Slow Read command (0x03). Valid for ECP5.
-mot	Generate Motorola 32-bit hex file. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LFE3-17EA-XXFN484` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default file name is the same name as the input file. Motorola files should end with the *.exo extension. **-compress** <on | off>: Specifies compression (optional). Default is off. **-verifyid**: On: Include the Verify ID frame. Off: Replaces Verify ID frame with NOOP (optional). Default is off. **-frequency**: Specifies the frequency of the master bitstream. Possible frequencies valid for LatticeEC/P, LatticeECP2/M, LatticeECP3, and LatticeXP/2 devices only are: 2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60, and 130 MHz. Possible frequencies valid for for MachXO2, MachXO3D, and MachXO3L devices only are: 2.1, 2.5, 3.2, 4.3, 5.5, 7, 8.3, 9.2, 10, 13, 15, 20, 27, 30, 33, 38, 44, 53, 66, 89, and 133 MHz. Possible frequencies valid for for ECP5 devices only are: 2.4, 3.2, 4.1, 4.8, 6.5, 8.2, 9.7, 12.9, 15.5, 16.3, 19.4, 20.7, 25.8, 31, 34.4, 38.8, 44.3, 51.7, 62, 77.5, 103.3, and 155 MHz. **-crc** <frame | global>: The "frame" option includes the CRC checking for each data frame. The "global option disables the frames CRC but still calculates the global CRC at the end of the configuration data. Default uses the settings of the input file. Valid for ECP5, LatticeECP2/M, LatticeECP2S/SM, LatticeECP3, MachXO3D, MachXO3L, MachXO2, and LatticeXP/2 devices only. **-mirror**: Flips each byte. Default: bytes are not flipped. Optional. **-header**: Retains the input file header in the output file. Default: Replaces header with all ones (0xFF). Optional. **-secure** <on | off>: Specifies program security (optional). On: Programs the Security Fuse (blocks read back). Off: Does not program the Security Fuse (allows read back).Default uses the settings of the input file.

Option	Description
-mot (cont.)	**-overwriteues on**: Overwrites (or replaces) the USERCODE value contained in the input file with the hex value passed in. **Note**: The encryption software patch must be installed before using the following **-encryption -key** and **-config_mode** commands. **-encryption -key hex** <Valid Key Value>: Encrypts the input file using the -key 128-bit encryption key. **-config_mode** <slave_scm | slave_pcm | jtag | spi | spim | master_pcm | slave_spi>]: Selects the appropriate configuration mode for the bitstream encryption. Valid for LatticeECP2S/MS and LatticeECP3 devices only. **-address** <hex address> Specifies the starting address of the memory device to store the input file. **-fast**: Using this option, the FPGA will issue the Fast Read command (0x0B) to the SPI Flash, instead of the Slow Read command (0x03). Valid for ECP5.
-xtek	Generate Tektronix Extended hex file **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LFE3-17EA-XXFN484` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Extended Tektronix files should end with the *.xtek extension. **-compress** <on | off>: Specifies compression (optional). Default is off. **-verifyid**: On: Include the Verify ID frame. Off: Replaces Verify ID frame with NOOP (optional). Default is off. **-frequency**: Specifies the frequency of the master bitstream. Possible frequencies valid for LatticeEC/P, LatticeECP2/M, LatticeECP3, and LatticeXP/2 devices only are: 2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60, and 130 MHz. Possible frequencies valid for for MachXO2, MachXO3D, and MachXO3L devices only are: 2.1, 2.5, 3.2, 4.3, 5.5, 7, 8.3, 9.2, 10, 13, 15, 20, 27, 30, 33, 38, 44, 53, 66, 89, and 133 MHz. Possible frequencies valid for for ECP5 devices only are: 2.4, 3.2, 4.1, 4.8, 6.5, 8.2, 9.7, 12.9, 15.5, 16.3, 19.4, 20.7, 25.8, 31, 34.4, 38.8, 44.3, 51.7, 62, 77.5, 103.3, and 155 MHz. **-crc** <frame | global>: The "frame" option includes the CRC checking for each data frame. The "global option disables the frames CRC but still calculates the global CRC at the end of the configuration data. Default uses the settings of the input file. Valid for ECP5, LatticeECP2/M, LatticeECP2S/SM, LatticeECP3, MachXO3D, MachXO3L, MachXO2, and LatticeXP/2 devices only.

Option	Description
-xtek (cont.)	**-mirror**: Flips each byte. Default: bytes are not flipped. Optional. **-header**: Retains the input file header in the output file. Default: Replaces header with all ones (0xFF). Optional. **-secure** <on | off>: Specifies program security (optional). On: Programs the Security Fuse (blocks read back). Off: Does not program the Security Fuse (allows read back).Default uses the settings of the input file. **overwriteues on**: Overwrites (or replaces) the USERCODE value contained in the input file with the hex value passed in. **Note**: The encryption software patch must be installed before using the following **-encryption -key** and **-config_mode** commands. **-encryption -key hex** <Valid Key Value>: Encrypts the input file using the -key 128-bit encryption key. **-config_mode** <slave_scm | slave_pcm | jtag | spi | spim | master_pcm | slave_spi>]: Selects the appropriate configuration mode for the bitstream encryption. Valid for LatticeECP2S/MS and LatticeECP3 devices only. **-address** <hex address> Specifies the starting address of the memory device to store the input file. **-fast**: Using this option, the FPGA will issue the Fast Read command (0x0B) to the SPI Flash, instead of the Slow Read command (0x03). Valid for ECP5.
-svf	Generate SVF file for a single device. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LCMXO2-256ZE-XXMG64` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default file name is the same name as the input file. SVF files should end with the *.svf extension. -**op** Indicates the operation. The operation list varies depending on the device family. Use the Deployment Tool GUI to create the command line. **-nocomments**: Omit Header and Comments from SVF file. Optional. Default includes Header and Comments. **-revd**: Generate Rev D standard SVF file. Optional. Default generates Lattice Extended SVF file (not supported by all third party tools). **-runtest**: Uses RUNTEST format from SVF revision C. Optional. Default is revision D SVF file. **-skipverify:** Generates an SVF without the verify portion. Optional. **-reset**: Writes RESET in the SVF file. Optional. **-maxdata** [<8 | 16 | 32 | 64 | 128 | 256>]: Specifies the maximum data size per row (in Kbits). Optional.

Option	Description
-svfchain	Generate SVF file for chain of devices. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default file name is the same name as the input file. SVF files should end with the *.svf extension. **-nocomments**: Omit Header and Comments from SVF file. Optional. Default includes Header and Comments. **-revd**: Generate Rev D standard SVF file. Optional. Default generates Lattice Extended SVF file (not supported by all third party tools). **-runtest**: Uses RUNTEST format from SVF revision C. Optional. Default is revision D SVF file. **-skipverify:** Generates an SVF without the verify portion. Optional. **-reset**: Writes RESET in the SVF file. Optional. **-maxdata** [<8 | 16 | 32 | 64 | 128 | 256>]: Specifies the maximum data size per row (in Kbits). Optional.
-stp	Generate Standard Test and Programming Language (STAPLE) file. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LCMXO2-256ZE-XXMG64` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.stp extension. -**op** Indicates the operation. The operation list varies depending on the device family. Use the Deployment Tool GUI to create the command line. **-nocompress:** < ON | OFF >: Generates a compressed or uncompressed STAPL file. Default is ON. **-print** < ON | OFF >: Generates a STAPL file with all the comment statements. Default is OFF. **-skipverify**: < ON | OFF >: Generates a STAPL file without the Verify portion. Optional.

Option	Description
-stpchain	Generate Standard Test and Programming Language (STAPL) file for a chain of devices. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.stp extension. **-nocompress:** < ON | OFF >: Generates a compressed or uncompressed STAPL file. Default is ON. **-print** < ON | OFF >: Generates a STAPL file with all the comment statements. Default is OFF. **-skipverify**: < ON | OFF >: Generates a STAPL file without the Verify portion. Optional.
-ate	Generate ATE file. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.tst, *.gr, *.pcf, or *.asc extension, descending type. **-type**: <tst | gr | hp3070 | hp3065 | t1800 | t200> **-ispdcd**: Generates vectors in ispDCD Format. **-skipverify**: For Erase, Program, and Verify Operations, Skip Verify. **-headerfile** <file path>: Specifies a customer header file. **-splitfile** [-init] [-vectors <# vectors/file>] [-splitatlow]: Split each file at active Low. Default: Split each file at active high.
-fullvme	Generate JTAG Full VME Embedded file. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.vme extension. **-maxdata** <8 | 16 | 32 | 64 | 128 | 256>: Maximum Memory Allocation Size Per Row of Data (Kbytes). Optional. **-hex**: Generate a Hex (.c) file along with the VME file. Optional. **-nocompress**: Do not compress data. Optional. **-compact**: Compact VME file. Optional. **-fixedpulse**: Generate a fixed pulse width VME file. Optional. **-verifyues**: Test the USERCODE. Program the device if USERCODE fails verify. Optional. **-noheader**: Omit header from VME file. Optional. **-comment**: Include comments in VME file. Optional.

Option	Description
-slimvme	Generate JTAG Slim VME Embedded file. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-ofa** "<Algorithm File Path>": Programming Algorithm file. Optional. Default use input with *.algo.vme. **-ofd** "<Data File Path>": Programming Data file. Optional. Default uses input with *.data.vme. **-nocompress**: Do not compress data. Optional. **-hex**: Generate a Hex (.c) file along with the Slim VME file. Optional.
-sspi	Generate Slave SPI embedded file. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-ofa** "<Algorithm File Path>": Programming Algorithm file. Optional. Default use input with *.algo.vme. **-ofd** "<Data File Path>": Programming Data file. Optional. Default uses input with *.data.vme. **-nocompress**: Do not compress data. Optional. **-hex**: Generate a Hex (.c) file along with the Slave SPI Embedded file. Optional. -**op** Indicates the operation. This is required when the input file is JEDEC. When the input file is an XCF, the XCF contains the operation.
-i2c	Generate I2C embedded file. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-ofa** "<Algorithm File Path>": Programming Algorithm file. Optional. Default use input with *.algo.vme. **-ofd** "<Data File Path>": Programming Data file. Optional. Default uses input with *.data.vme. **-nocompress**: Do not compress data. Optional. **-hex**: Generate a Hex (.c) file along with the VME file. Optional. -**op** Indicates the operation. This is required when the input file is JEDEC. When the input file is an XCF, the XCF contains the operation. **-address** <80 | (User Specified)> Specifies the slave I2C address. **-comment:** Include comments in I2C embedded file. Optional. **-fixedpulse**: Generate a fixed pulse width I2C embedded file. Optional.

Option	Description
-cpu	Generates sysCONFIG CPU Embedded file. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with the extension of the selected **-type**. **-type** <cpu | c | hex | txt>: Indicates whether file is in binary embedded (cpu), C-code (c), Intel Hex (hex), or Text debug only (txt) format. **-verify:** Generates a sysCONFIG Embedded file that includes a verify operation. **-comment**: Include comments in sysCONFIG Embedded file. Optional. **-nocompress**: Do not compress data. Optional. **-mirror**: Flips each byte. Default: bytes are not flipped. Optional.
-boot	Generate Dual Boot Hex file. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LFE3-17EA-XXFN484` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-golden** "<Golden File Path>" **-primary** "<Primary File Path>": Merges the Golden, Primary, and a JUMP command into one Hex file. The address location of the Golden, Primary, and JUMP command depends on the files sizes and the target device. The Deployment Toll will record the address location of each in the Status window and log file. **-format** < intel | motorola | xtek >: Specifies the output format as Intel, Motorola, or Tektronix Extended Hex. The default format is Intel Hex. Optional. Default uses the input file name with *.mcs, *exo, or *.xtek extension, depending on format type selected. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.mcs, *exo, or *.xtek extension, depending on format type selected. **-flashsize:** Specifies size in Mb. Possible PROM sizes are 1, 2, 4, 8, 16, 32, 64, 128, 256, and 513 (512Mb). Default: 1.

Option	Description
-boot (cont.)	**-protect:** Optional. Inserts the Golden File at the beginning of the upper half of the SPI flash. This allows the upper half of the SPI flash to be write-protected, which protects the Golden File from accidental erase or reprogram. **-mirror**: Flips each byte. Default: bytes are not flipped. Optional. **-header**: Retains the input file header in the output file. Default: Replaces header with all ones (0xFF). Optional. **-optimize**: Uses the files size of the input files to determine SPI flash sector addresses for each input file. Use this option to minimize wasted memory space. The default is to use worst case file size fo the target device, which is an uncompressed file with all EBR and PCS, depending on the family. Using the worst case will ensure that all files will always be located at the same address even if any of the files/designs are updated at a later date. **-asc** <1 | 2 | 3 | 4 | 5 | 6 | 7 | 8> Indicates the number of ASC devices connected to the MachXO2 or Platform Manager 2 device. The maximum number that can be used with the MachXO2 device is 8. The maximum number that can be used with the Platform Manager 2 device is 7. The input JEDEC file and the ASC Hex files will be merged into one Hex file that can be programmed into an external SPI Flash device for Dual Boot support. Optional. **-ascfile** "<Data File Path>": Specifies the ASC Hex file path and name. If the file path includes spaces, enclose the path in quotes. There must be one "-ascfile" and path for each ASC specified by the "-asc" Optional. <**-fast** | **-dualio** | **-quad 1**>: With the -fast option, the FPGA will issue the Fast Read command (0x0B) to the SPI Flash. The -dualio option must be used if a Dual I/O SPI Flash is used. The -quad 1 option must be used if a Quad I/O SPI Flash is used. Only one of these options (-fast, -dualio, -quad 1) can be used. Without any of these options, the FPGA uses the Slow Read command (0x03). Valid for ECP5. **-encryption -key hex** <Valid Key Value>: Encrypts the input file using the -key 128-bit encryption key.
-jed	Generates JEDEC file. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LCMXO2-256ZE-XXMG64` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.jed extension. **-overwriteues** <**on**>: Overwrites (or replaces) the USERCODE value contained in the input file with the hex value passed in. The Checksum option overwrites the USERCODE with the JEDEC file fuse checksum value. The Disable option omits the USERCODE from the JEDEC file. **Note**: The encryption software patch must be installed before using the following **-encryption -key** commands. **-encryption -key hex** <Valid Key Value>: Encrypts the input file using the -key 128-bit encryption key.

Option	Description
-advanced	Generates Advanced SPI Flash Hex file. **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LFE3-17EA-XXFN484` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-if** <Data File Path>: Indicates the input data file full path and file name. If the file path includes spaces, enclose the path in quotes. **-format** < intel | motorola | xtek >: Specifies the output format as Intel, Motorola, or Tektronix Extended Hex. The default format is Intel Hex. Optional. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.mcs. *.exo, or *.xtek extension, depending on the format selected. **-flashsize:** Specifies size in Mb. Possible PROM sizes are 512 (512Kb), 1, 2, 4, 8, 16, 32, 64, 128 , 256, and 513 (512Mb). Default: 1. **-mirror**: Flips each byte. Default: bytes are not flipped. Optional. **-header**: Retains the input file header in the output file. Default: Replaces header with all ones (0xFF). Optional. **-userdata** <*1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | *14* | *15* | *16*> Number of data files. **-userdata**: Flips each byte of the user data files. Default: bytes are not flipped. Optional. **-userfile** "<Data File Path>" Indicates the input user data file path and file name. Must be specified for each user data file (**-userdata**). **-address** <hex address value>: SPI flash address location for the user data file. Must be specified for each user data file (**-userdata**). **-optimize**: Uses the files size of the input files to determine SPI flash sector addresses for each input file. Use this option to minimize wasted memory space. The default is to use worst case file size fo the target device, which is an uncompressed file with all EBR and PCS, depending on the family. Using the worst case will ensure that all files will always be located at the same address even if any of the files/designs are updated at a later date. **-ice**: iCE40 warm boot cold boot. **-warm** <1 | 2 | 3 | 4>: Optional. Default is cold boot. The **-warm** selects warm boot. The number selects the first boot configuration file. **-patterns** <2,3,4>: Specifies the number of iCE40 configuration files. **-bootfile** "<Data File Path>": Indicates the input configuration file path and name. Must be specified for each configuration file (**-patterns**). **-address** <hex address value>: SPI flash address location for the configuration file. Must be specified for each configuration file (**-patterns**). <**-fast** | **-dualio** | **-quad 1**>: With the -fast option, the FPGA will issue the Fast Read command (0x0B) to the SPI Flash. The -dualio option must be used if a Dual I/O SPI Flash is used. The -quad option must be used if a Quad I/O SPI Flash is used. Only one of these options (-fast, -dualio, -quad 1) can be used. Without any of these options, the FPGA uses the Slow Read command (0x03). Valid for ECP5.

Option	Description
-advanced (cont.)	**-multi** <1 | 2 | 3 | 4> Optional. The -multi option selects Multiple Boot. The number selects the number of alternate files. Must be used in conjunction with Dual Boot, since dual boot is a subset of Multiple Boot feature. The Golden file must also be selected. The data file passed in using the -if option is considered the Primary file. **-golden** "<Data File Path>" Indicates the Golden file path and file name. **-altfile** "<Data File Path>" Indicates the alternate file path and name. Must specify the alternate file for each number following the -multi option. **-address** <hex address value> SPI Flash address location for the alternate file. Must be specified for each alternate file (-altfile). **-next** <primary | alt1 | alt2 | alt3 |alt4> Indicates which will be the next pattern to boot from when the PROGRAMN pin is toggle, or the Refresh command is issued. Must be specified for each alternate file (-altfile).
-merge	**-ifdev1** "<Data File Path>" Indicates the input dta file full path and file name for the first device in the chain. If the file path includes spaces. **-ifdev2** "<Data File Path>" "Second device in the chain." **-format** < intel | motorola | xtek >: Specifies the output format as Intel, Motorola, or Tektronix Extended Hex. The default format is Intel Hex. Optional. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.mcs. *.exo, or *.xtek extension, depending on the format selected. **-mergeformat** <intelligent | combine> Intelligent merge will set the appropriate bits in the bitstream to support sysCONFIG Daily Chaining. Combine will merge the two files without changing any bits in the bitstreams. **-frequency**: Specifies the frequency of the master bitstream. Possible frequencies valid for LatticeEC/P, LatticeECP2/M, LatticeECP3, and LatticeXP/2 devices only are: 2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60, and 130 MHz. Possible frequencies valid for for MachXO2 and MachXO3L devices only are: 2.1, 2.5, 3.2, 4.3, 5.5, 7, 8.3, 9.2, 10, 13, 15, 20, 27, 30, 33, 38, 44, 53, 66, 89, and 133 MHz. Possible frequencies valid for for ECP5 devices only are: 2.4, 3.2, 4.1, 4.8, 6.5, 8.2, 9.7, 12.9, 15.5, 16.3, 19.4, 20.7, 25.8, 31, 34.4, 38.8, 44.3, 51.7, 62, 77.5, 103.3, and 155 MHz. **-scoutput** <dout | qout>: For LatticeSC/M devices, indicates if the output pin is Dout or Qout. **-mirror**: Flips each byte. Default: bytes are not flipped. Optional. **-header**: Retains the input file header in the output file. Default: Replaces header with all ones (0xFF). Optional.

Option	Description
-bsm	Generates Application Specific BSDL file. **-ifd** "<Data File Path>": Indicates the input data file full path and file name.If the file name includes spaces ... **-ifb** "<BSDL File Path>": Indicates the input BSDL file full path and file name. If the file path includes spaces ... **-dev** <Device Name>: Indicates the name of the device. Optional. The device name must be the full device name with the performance grade replaced with two X's, and minus the industry rating or the code name. Example: `LCMXO2-256ZE-XXMG64` (with performance grade replaced with two X's). The device name will default to the device name in the data file. If a device name cannot be found in the file, it will return an error unless the name has been given with the -dev option. **-of** "<Data File Path>": Indicates the output file path. If the file path includes spaces, enclose the path in quotes. Optional. Default uses the input file name with *.??? extension. **-convertbidi**: Converts bidirectional I/Os to input or output only, based on the user's design. Default is to keep al bidirectional I/Os as bidirectional. Optional.
-jed2hex	Generates an ASCII Raw Hex file.
-jed2bin	Generates a Binary Raw Hex file.

Examples The following are DDTCMD examples.

Tester Examples:

SVF - Single Device:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -svfsingle -if
"C:/example/lfxp2_05ef256.jed" -dev LFXP2-5E -op "FLASH
Erase,Program,Verify" -of  "C:/ example/
lfxp2_05ef256.svf"
```

SVF - JTAG Chain:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -svfchain -if
"C:/example/demo.xcf" -of  "C:/ example/demo.svf"
```

STAPL - Single Device:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -stpsingle -if
"C:/example/ isppac_powr1220at8.jed" -dev ispPAC-
POWR1220AT8 -op "Erase,Program,Verify" -of  "C:/ example/
isppac_powr1220at8.svf"
```

STAPL - JTAG Chain:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -stpchain -if
"C:/example/demo.xcf" -of  "C:/example/demo.svf"
```

ATE - Generic Vector Format:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type tst -of " C:/example/demo.tst"
```

ATE - GenRad:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type gr -of " C:/example/demo.gr"
```

ATE - HP3070:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type hp3070 -of " C:/example/
demo.pcf"
```

ATE - HP3065:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type hp3065 -of " C:/example/
demo.pcf"
```

ATE - Teradyne 1800:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type t1800 -of " C:/example/demo.asc"
```

ATE - Teradyne L200/300:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -ate -if " C:/
example/demo.xcf " -type t200 -of " C:/example/demo.asc"
```

Embedded System Examples:

JTAG Full VME Embedded:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -fullvme -if
"C:/example/demo.xcf" -of "C:/example/demo.vme"
```

JTAG Slim VME Embedded:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -slimvme -if
"C:/example/demo.xcf" -ofa "C:/ example/demo_algo.vme" -
ofd "C:/ example/demo_data.vme"
```

Slave SPI Embedded:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -sspi -if "C:/
example/demo.xcf" -ofa "C:/example/demo_algo.sea" -ofd
"C:/example/demo_data.sed"
```

I2C Embedded:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -i2c -if "C:/
example/demo.xcf" -ofa "C:/example/demo_algo.iea" -ofd
"C:/example/demo_data.iea"
```

sysCONFIG (CPU) Embedded - Binary:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -cpu -if "C:/
example/demo.xcf" -type cpu -of "C:/example/
demo_algo.cpu"
```

sysCONFIG (CPU) Embedded - C-Code:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -cpu -if "C:/
example/demo.xcf" -type c -of "C:/example/demo_algo.c"
```

sysCONFIG (CPU) Embedded - Intel Hex:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -cpu -if "C:/
example/demo.xcf" -type hex -of "C:/example/
demo_algo.hex"
```

sysCONFIG (CPU) Embedded - Text (Debug Only):

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -cpu -if "C:/
example/demo.xcf" -type txt -of "C:/example/
demo_algo.txt"
```

External Memory Examples:

Hex Conversion - Intel Hex:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -int -dev LFE2-
6E -if "C:/example/lfec2_06ef256.bit" -of "C:/example/
lfec2_06ef256.mcs"
```

Hex Conversion - Motorola Hex:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -mot -dev LFE2-
6E -if "C:/example/lfec2_06ef256.bit" -of "C:/example/
lfec2_06ef256.exo"
```

Hex Conversion - Extended Tektronix Hex:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -xtek -dev LFE2-
6E -if "C:/example/lfec2_06ef256.bit" -of "C:/example/
lfec2_06ef256.xtek"
```

Dual Boot - Intel Hex:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -boot -dev LFE3-
95E -golden "C:/example/lfec3_095_golden.bit" -primary
"C:/example/lfec3_095.bit" -format int -flashsize 64 -of
"C:/example/lfec3_095_dual_boot.mcs"
```

Dual Boot - Motorola Hex:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -boot -dev LFE3-
95E -golden "C:/example/lfec3_095_golden.bit" -primary
"C:/example/lfec3_095.bit" -format mot -flashsize 64 -of
"C:/example/lfec3_095_dual_boot.exo"
```

Dual Boot - Extended Tektronix Hex:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -boot -dev LFE3-
95E -golden "C:/example/lfec3_095_golden.bit" -primary
"C:/example/lfec3_095.bit" -format xtek -flashsize 64 -of
"C:/example/lfec3_095_dual_boot.xtek"
```

File Conversion Examples:

Bitstream - Binary Bitstream:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -bit -dev LFXP2-
5E -if "C:/example/lfxp2_05ef256.jed" -of "C:/example/
lfxp2_05ef256.bit"
```

Bitstream - ASCII Bitstream:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd" -oft -rbt -dev LFE2-
6E -if "C:/example/lfec2_06ef256.bit" -of "C:/example/
lfec2_06ef256.rbt"
```

JEDEC - Hex and JEDEC - Binary:

```
"C:/lscc/diamond/3.5/bin/nt/ddtcmd -oft <jed2hex|jed2bin>
[-dev <Device Name>] -if "<Data File Path>" [-of "<Data
File Path>"]
```

Return Codes

Code	Definition
0	CMD_SUCCESS
-1	ERROR_LOG_FILE
-2	OUT_MEMORY
-3	CANNOT_WRITE_DIRECTORY
-4	ERROR_CONFIGURATION_SETUP
-5	DEVICE_NAME_NOT_FOUND
-6	FILE_NOT_FOUND
-7	FILE_NOT_VALID
-8	UNKNOWN_OPERATION
-9	TOO_MANY_CHIPS
-10	FILE_NOT_JEDEC
-11	ERROR_OUTPUT_FILE
-12	OPERATION_NOT_SUPPORT
-13	FILE_NAME_ERROR
-14	FILE_READ_ERROR
-15	ERROR_BUILD_SVP_FILE
-16	ERROR_BUILD_BJD_FILE
-17	ERROR_BUILD_SVF_FILE
-18	CANNOT_WRITE_LOG_FILE
-19	MISSING_START_FILE
-20	UNKNOWN_DEVICE
-21	ERRROR_BUILD_BIT_FILE
-22	MULTIPLE_DEVICES_FOUND
-23	DEVICE_NOT_MATCH_FILE
-24	ERROR_COMMAND_LINE_SYNTAX
-25	ERROR_BUILD_ISC_FILE
-26	ERROR_BUILD_SPLIT_BITSTREAM_FILE
-27	ERROR_BUILD_MERGE_BITSTREAM_FILE
-28	ERROR_INPUT_FILE
-29	ERROR_BUILD_BITSTREAM_FILE
-30	ERROR_BUILD_JED_FILE

Code	Definition
-31	ERROR_BUILD_PROM_FILE
-32	DEVICE_NOT_SUPPORTED

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Running I/O SSO Analysis from the Command Line

The **I/O SSO Analysis Report** process in the Diamond environment can also be run through the command line using the SSO Analysis (**ssoana**) program. SSO analysis describes the noise on signal and supply lines caused by a large number of output drivers that are switching at the same time. The SSO analysis report helps ensure that your I/O plan meets the power integrity requirements of the PCB design.

Subjects included in this topic:

- Running SSOANA
- Command Line Syntax
- SSOANA Options
- Examples

Running SSOANA The SSOANA program generates an SSO (Simultaneous Switching Output) analysis report (.sso) based on the SSO preferences you specified in your LPF file.

- To run SSOANA, type **ssoana** on the command line with all necessary options:

  ssoana -d LFECP6E -p TQFP144 -o mydesign.sso mydesign.ncd mydesign.lpf

There are several command line options that give you control over the way SSOANNA outputs analysis files. Please refer to the rest of the subjects in this topic for more details. See examples.

Command Line Syntax **ssoana** -d <device> -p <package> -o <report_file> <design_file> <preference_file> [-ps <pref_source>]

SSOANA Options

The table below contains descriptions of all valid options for SSOANA.

Table 912: SSOANA Command Line Options

Option	Description
-d <device>	Specifies the device name.
-p <package>	Specifies the package name.
-o <report_file>	Specifies the output I/O SSO Analysis report file name.
<design_file>	Specifies the input physical design file (NCD).
<preference_file>	Specifies the input logical preference (LPF) file name.
-ps <pref_source>	Specifies the preference source. Options are **lpf** or **null**. If no -ps option is specified, the preference will be lpf and ncd and ncd will cover lpf. If lpf is specified, the preference source will be uniquely the lpf file.

Examples Following are a few examples of SSOANA command lines and a description of what each does.

Example 1 The following command generates an output (-o option) I/O SSO analysis report file (ddrg_lvr71.sso) for a LatticeECP2-12E device in a 256-ball, fine pitch Ball Grid Array package from input project NCD and LPF files:

ssoana -d LFE2-12E -p FPBGA256 ddrg_lvr71.ncd -o ddrg_lvr71.sso ddrg_lvr71.lpf

Example 2 The following command line specifies an additional “-ps lpf” option, so that SSOANA only reads preferences, such as I/O assignments, from LPF file. You can use this option to calculate the SSO of different I/O planning without replacement:

ssoana -d LFE2-12E -p FPBGA256 ddrg_lvr71.ncd -o ddrg_lvr71.sso ddrg_lv71.lpf -ps lpf

See Also ▶“Command Line Data Flow” on page 2467

▶ “Command Line Program Overview” on page 2464

Running Various Utilities from the Command Line

The command line utilities described in this section are not commonly used by command line users, but you often see them in the auto-make log when you run design processes in the Diamond environment. Click each link below for its function, syntax, and options.

- synpwrap
- htmlrpt

Note

For information on commonly-used FPGA command line tools, see "Command Line Reference Guide" on page 2464.

synpwrap The **synpwrap** command line utility (wrapper) is used to manage Synplicity Synplify and Synplify Pro synthesis programs from the Diamond environment processes: **Synplify Synthesize Verilog File** or **Synplify Synthesize VHDL File**.

The **synpwrap** utility can also be run from the command line to support a batch interface. For details on Synplify see the ispLEVER online help. The **synpwrap** program drives **synplify** or **synplify_pro** programs with a Tcl script file containing the synthesis options and file list.

Note

This section supersedes the "Process Optimization and Automation" section of the *Synplicity Synplify and Synplify Pro for Lattice User Guide*.

This section illustrates the use of the **synpwrap** program to run Synplify or Synplify Pro (Synplify/Pro) for Lattice synthesis scripts from the command line. For more information on synthesis automation of Synplify or Synplify Pro, see the "User Batch Mode" section of the *Synplicity Synplify and Synplify Pro for Lattice User Guide*. If you use the Lattice OEM Synplify/Pro, the Lattice OEM license requires that the command line executables: **synplify** and **synplify_pro** be run by the Lattice "wrapper" program, **synpwrap**.

Command Line Syntax **synpwrap** [**-log** <*log_file*>] [**-nolog**] [**-int** <*command_file*>] [**-gui**] [**-int** <*project_file*> | **-prj** <*project_file*>] [**-dyn**] [**-notoem**] [**-oem**] [**-notpro**] [**-pro**] [**-rem**] [**-scriptonly** <*script_file*>] **-e** <*command_file*> **-target** <*device_family*> **-part** <*device_name*> [**-options** <*arguments*>]

Table 913: SYNPWRAP Command Line Options

Option	Description
-log <*log_file*>	Specifies the log file name.
-nolog	Does not print out the log file after the process is finished.
-options <*arguments*>	Passes all arguments to Synplify/Pro. Ignores all other options except -notoem/-oem and -notpro/-pro. The -options switch must follow all other synpwrap options.
-prj <*project_file*>	Runs Synplify or Synplify Pro using an external prj Tcl file instead of the Diamond command file.
-rem	Does not automatically include Lattice library files.

Table 913: SYNPWRAP Command Line Options

Option	Description
-e <*command_file*>	Runs the batch interface based on a Diamond generated command file. The synpwrap utility reads <*project*>.cmd with its command line to obtain user options and creates a Tcl script file.
-gui	Invokes the Synplify or Synplify Pro graphic user interface.
-int <*command_file*>	Enables the interactive mode. Runs Synplify/Pro UI with project per command file.
-dyn	Brings the Synplify installation settings in the Diamond environment.
-notoem	Does not use the Lattice OEM version of Synplify or Synplify Pro.
-oem	Uses the Lattice OEM version of Synplify or Synplify Pro.
-notpro	Does not use the Synplify Pro version.
-pro	Uses the Synplify Pro version.
-target <*device_family*>	Specifies the device family name.
-part <*device_name*>	Specifies the device. For details on legal <*device_name*> values.
-scriptonly <*script_file*>	Generates the Tcl file for Synplify or Synplify Pro. Does not run synthesis.

Example Below shows a synpwrap command line example.

```
synpwrap -rem -e prep1 -target LATTICE-EC -part LFEC20E
```

htmlrpt This command line tool is a report converter that converts the text format report files into HTML report files.

Command Line Syntax **htmlrpt** [**-dly** <*dlyfile*>] [**-mrp** <*mrpfile*>] [**-pad** <*padfile*>] [**-par** <*parfile*>] [**-mtwr** <*mtwrfile*>] [**-ptwr** <*ptwrfile*>] [**-bgn** <*bgnfile*>] [**-jdr** <*bgnfile*>][**-prm** <*prmfile*>] [**-info** <*infofile*>] [**-db** <*databasefile*>] [**-syn** <*synfile*>] [**-lse** <*synfile*>] [**-sso** <*ssofile*>]<*design*>

Table 914: HTMLRPT Command Line Options

Option	Description
-dly <*dlyfile*>	Specifies the delay file.
-mrp <*mrpfile*>	Specifies the map report file.
-pad <*padfile*>	Specifies the pad specification file.
-par <*parfile*>	Specifies the par report file.
-mtwr <*mtwrfile*>	Specifies the map trace report file.

Table 914: HTMLRPT Command Line Options

Option	Description
-ptwr <*ptwrfile*>	Specifies the par trace report file.
-bgn <*bgnfile*>	Specifies the bitgen report file.
-jdr <*bgnfile*>	Specifies the bitgen report file for XP/XP2.
-prm <*prmfile*>	Specifies the promgen report file.
-info <*infofile*>	Specifies the promgen report file.
-db <*databasefile*>	Specifies the database report file.
-syn <*synfile*>	Specifies the synthesis and NGDBUILD report file.
-lse<*synfile*>	Specifies the Lattice synthesis report file.
-sso <*ssofile*>	Specifies the I/O SSO Analysis report file.

See Also ▶"Command Line Data Flow" on page 2467

▶ "Command Line Program Overview" on page 2464

Using Command Files

This section describes how to use command files.

Creating Command Files The command file is an ASCII file containing command arguments, comments, and input/output file names. You can use any text editing tool to create or edit a command file, for example, **vi**, **emacs**, **Notepad**, or **Wordpad**.

Here are some guidelines when you should observe when creating command files:

- Arguments (executables and options) are separated by space and can be spread across one or more lines within the file.
- Place new lines or tabs anywhere white space would otherwise be allowed on the UNIX or DOS command line.
- Place all arguments on the same line, or one argument per line, or any combination of the two.
- There is no line length limitation within the file.
- All carriage returns and other non-printable characters are treated as space and ignored.
- Comments should be preceded with a # (pound sign) and go to the end of the line.

Command File Example This is an example of a command file:

```
#command line options for par for design mine.ncd
-a -n 10
-w
```

```
-l 5
-s 2 #will save the two best results
/home/users/jimbob/designs/mine.ncd
#output design name
/home/users/jimbob/designs/output.dir
#use timing preference file
/home/users/jimbob/designs/mine.prf
```

Using the Command File The –f Option Use the **–f** option to execute a command file from any command line tool. The **–f** option allows you to specify the name of a command file that stores and then executes commonly used or extensive command arguments for a given FPGA command line executable tool. You can then execute these arguments at any time by entering the UNIX or DOS command line followed by the name of the file containing the arguments. This can be useful if you frequently execute the same arguments each time you perform the command, or if the command line becomes too long. This is the recommended way to get around the DOS command line length limitation of 127 characters. (Equivalent to specifying a shell Options file.)

The **–f** indicates fast startup, which is performed by not reading or executing the commands in your .cshrc | .kshrc | .shrc (C-shell, Korn-shell, Bourne-shell) file. This file typically contains your path information, your environment variable settings, and your aliases. By default, the system executes the commands in this file every time you start a shell. The **–f** option overrides this process, discarding the 'set' variables and aliases you do not need, making the process much faster. In the event you do need a few of them, you can add them to the command file script itself.

Command File Usage Examples You can use the command file in two ways:

- To supply all of the command arguments as in this example:

par -f <command_file>

where:

<command_file> is the name of the file containing the command line arguments.

- To insert certain command line arguments within the command line as in the following example:

par -i 33 -f placeoptions -s 4 -f routeoptions design_i.ncd design_o.ncd

where:

placeoptions is the name of a file containing placement command arguments.

routeoptions is the name of a file containing routing command arguments.

Using Command Line Shell Scripts

This topic discusses the use of shell scripts to automate either parts of your design flow or entire design flows. It also provides some examples of what you can do with scripts. These scripts are UNIX-based; however, it is also possible to create similar scripts called batch files for PC but syntax will vary in the DOS environment.

Creating Shell Scripts A UNIX shell script is an ASCII file containing commands targeted to a particular shell that interprets and executes the commands in the file. For example, you could target Bourne Shell (**sh**), C-Shell (**csh**), or Korn Shell (**ksh**). These files also can contain comment lines that describe part of the script which then are ignored by the shell. You can use any text editing tool to create or edit a shell script, for example, **vi** or **emacs**.

Here are some guidelines when you should observe when creating shell scripts:

- It is recommended that all shell scripts with "#!" followed by the path and name of the target shell on the first line, for example, #!/bin/ksh. This indicates the shell to be used to interpret the script.
- It is recommended to specify a search path because oftentimes a script will fail to execute for users that have a different or incomplete search path. For example:

  PATH=/home/usr/lsmith:/usr/bin:/bin; export PATH
- Arguments (executables and options) are separated by space and can be spread across one or more lines within the file.
- Place new lines or tabs anywhere white space would otherwise be allowed on the UNIX command line.
- Place all arguments on the same line, or one argument per line, or any combination of the two.
- There is no line length limitation within the file.
- All carriage returns and other non-printable characters are treated as space and ignored.
- Comments are preceded by a # (pound sign) and can start anywhere on a line and continue until the end of the line.
- It is recommended to add exit status to your script, but this is not required.

```
# Does global timing meet acceptable requirement range?
    if [ $timing -lt 5  -o  $timing -gt 10 ]; then
         echo 1>&2 Timing \"$timing\" out of range
         exit 127
    fi
    etc...
    # Completed, Exit OK
    exit 0
```

Advantages of Using Shell Scripts Using shell scripts can be advantageous in terms of saving time for tasks that are often used, in

reducing memory usage, giving you more control over how the FPGA design flow is run, and in some cases, improving performance.

Scripting with DOS Scripts for the PC are referred to as batch files in the DOS environment and the common practice is to ascribe a .bat file extension to these files. Just like UNIX shell scripts, batch files are interpreted as a sequence of commands and executed. The COMMAND.COM or CMD.EXE (depending on OS) program executes these commands on a PC. Batch file commands and operators vary from their UNIX counterparts. So, if you wish to convert a shell script to a DOS batch file or vice-versa, we suggest you find a good general reference that shows command syntax equivalents of both operating systems.

Examples The following examples are examples of UNIX shell scripts written to run parts of the design flow.

Example 1 Using the Korn shell, this script runs to separate flows using pfu.edn and pio.edn inputs through place-and-route, respectively. It then runs the clock boosting on each flow, lists applicable files for analysis, and outputs error messages that may have resulted in an errors file.

```
#$!/bin/ksh
#
# clockboosting.script
#
PATH=/home/mjones/design_run:/usr/bin:/bin; export PATH
#
# First design flow
#
edif2ngd -l LatticeEC pfu.edn pfu.ngo
ngdbuild -a LatticeEC  pfu.ngo pfu.ngd
map -a LatticeEC pfu.ngd -o pfu.ncd test.prf
par -l 1 -n 1 -i 5 -c 0 pfu.ncd -w pfu.ncd test.prf
cst pfu.ncd test.prf
#
# Second design flow
#
edif2ngd -l LatticeEC pio.edn pio.ngo
ngdbuild -a LatticeEC pio.ngo pio.ngd
map -a LatticeEC pio.ngd -o pio.ncd test.prf
par -l 1 -n 1 -i 5 -c 0 pio.ncd -w pio.ncd test.prf
cst pio.ncd test.prf
ls *.grf *.cst > files
grep -i error *.Out *.Err >errors
```

Chapter 21

Tcl Command Reference Guide

Diamond software supports Tcl (Tool Command Language) scripting and provides extended Diamond Tcl commands that enable a batch capability for running tools in Diamond's graphical interface. The command set and the Tcl Console used to run it affords you the speed, flexibility and power to extend the range of useful tasks that the Diamond tools arealready designed to perform.

In addition to describing how to run Diamond's Tcl Console, this guide provides you with a reference for Tcl command line usage and syntax for all Diamond point tools within the graphical user interface so that you can create command scripts, modify commands, or troubleshoot existing scripts.

About the Diamond Tcl Scripting Environment The Lattice Diamond development software features a powerful script language system. The user interface incorporates a complete Tcl command interpreter. The command interpreter is enhanced further with additional Diamond-specific support commands. The combination of fundamental Tcl along with the commands specialized for use with Diamond allow the entire Diamond development environment to be manipulated.

Using the command line tools permits you to do the following:

- Develop a repeatable design environment and design flow that eliminates setup errors that are common in GUI design flows
- Create test and verification scripts that allow designs to be checked for correct implementation
- Run jobs on demand automatically without user interaction

The Diamond command interpreter provides an environment for managing your designs that are more abstract and easier to work with than using the core Diamond engines. The Diamond command interpreter does not prevent use of the underlying transformation tools. You can use either the TCL commands described in this section or you can use the core engines described in the Command Line Reference Guide.

Additional References If you are unfamiliar with the Tcl language you can get help by visiting the Tcl/tk web site at http://www.tcl.tk. If you already know how to use Tcl, see the Tcl Manual supplied with this software. For information

on command line syntax for running core tools that appear as Diamond processes, such as edif2ngd, map, par, ldbanno, and trce, see the "Command Line Reference Guide" on page 2464.

See Also "Running the Tcl Console" on page 2587

- "Launching the EPIC Tcl Console" on page 2588
- "Diamond Tool Tcl Command Syntax" on page 2593
- "Creating and Running Custom Tcl Scripts" on page 2590
- "Accessing Command Help in the Tcl Console" on page 2589
- Tcl Manual

Running the Tcl Console

The Diamond TCL Console environment is made available for your use in multiple different ways. In order to take full advantage of the FPGA development process afforded by the Diamond software you must gain access to the Diamond Tcl Console user interface.

On Windows In Windows 7 you can interact with the Tcl Console by any one of the following methods:

- To launch the Diamond GUI from the Windows Start menu, choose **Start > All Programs > Lattice Diamond > Lattice Diamond**.

  After the Diamond software loads you can click on the **TCL Console** tab. With the **TCL Console** tab active, you are able to start entering standard syntax TCL commands or the Diamond specific support commands.

- To launch the **TCL Console** independently from the Diamond GUI from the Windows Start menu choose **Start > All Programs > Lattice Diamond > Accessories > TCL Console**.

  A Windows command interpreter will be lauched that automatically runs the **TCL Console**.

- To run the interpreter from the command line, type the following:

  `c:/lscc/diamond/<version_number>/bin/nt/pnmainc`

  The Diamond **TCL Console** is now available to run.

- To run the interpreter from a Windows 7 PowerShell from the Windows Start menu choose **Start > All Programs > Accessories > Windows PowerShell > Windows PowerShell (x86)**.

  A PowerShell interpreter window will open. At the command line prompt type the following:

  `c:/lscc/diamond/<version_number>/bin/nt/pnmainc`

The Diamond **TCL Console** is now available to run.

Note

The arrangement and location of each of the programs in the Windows Start menu will differ depending on the version of Windows you are running.

On Linux In Linux operating systems you can interact with the Tcl Console by one of the following methods:

- To launch the Diamond GUI from the command line, type the following:

  ```
  /usr/local/diamond/<version_number>/bin/lin/diamond
  ```

  The path provided assumes the default installation directory and that Diamond is installed. After the Diamond software loads you can click on the **TCL Console** tab. With the **TCL Console** tab active, you are able to start entering standard syntax TCL commands or the Diamond specific support commands.

- To launch the **TCL Console** independently from the Diamond GUI from the command line, type the following:

  ```
  /usr/local/diamond/<version_number>/bin/lin/pnmainc
  ```

  The path provided assumes the default installation directory and that Diamond is installed, and that you have followed the Diamond for Linux installation procedures. The Diamond **TCL Console** is now ready to accept your input.

  The advantage of running the **TCL Console** from an independent command interpreter is the ability to directly pass the script you want to run to the Tcl interpreter. Another advantage is that you have full control over the Tk graphical environment, which allows you to create you own user interfaces.

See Also
- "Running the Tcl Console" on page 2587
- "Diamond Tool Tcl Command Syntax" on page 2593
- "Creating and Running Custom Tcl Scripts" on page 2590
- "Accessing Command Help in the Tcl Console" on page 2589
- Tcl Manual

Launching the EPIC Tcl Console

EPIC (Editor for Programmable Integrated Circuits) is a graphical application for viewing and configuring FPGAs. EPIC is a stand-alone tool that is run independently from the Diamond software.

The EPIC Tcl commands allow you to perform basic actions. The EPIC tool commands are described in the EPIC online help topic entitled “EPIC Tcl Commands”.

You can run EPIC tool Tcl commands using the EPIC Tcl Console (fpgac.exe).

- On Windows, from the command line, type the following:

  ```
  c:/lscc/diamond/<version_number>/bin/nt/fpgac
  ```

  The Epic **TCL Console** is now available to run.

- On Linux, from the command line, type the following:

  ```
  cd /usr/local/diamond/<version_number>/bin/lin
  source setupenv
  ./fpgac
  ```

  The path provided assumes the default installation directory and that Diamond is installed, and that you have followed the Diamond for Linux installation procedures. The Epic **TCL Console** is now ready to accept your input.

See Also ▶“Engineering Change Order Tcl Commands” on page 2630

Accessing Command Help in the Tcl Console

You can access command syntax help for all of the tools in the Tcl Console.

To access command syntax help in the Tcl Console:

1. In the prompt, type **help <tool_name>** and press **Enter** as shown below:

   ```
   help prj
   ```

   A list of valid command options appears in the Tcl Console. If you just type **help** in the console, you will be given a list of tools.

2. As directed in the window, type the name of the command or function for more details on syntax and usage. For the prj tool, for example, type and enter the following:

   ```
   prj_strgy
   ```

   A list of valid arguments for that function will appear.

Note

Although you can run Diamond’s core tools such as edif2ngd, map, par, ldbanno, and trce from the Tcl Console, the syntax for accessing help is different. For proper usage and syntax for accessing help for core tools, see the “Command Line Reference Guide” on page 2464.

See Also ▶“Running the Tcl Console” on page 2587

- “Diamond Tool Tcl Command Syntax” on page 2593
- “Creating and Running Custom Tcl Scripts” on page 2590

- "Running Tcl Scripts When Launching Diamond" on page 2592
- Tcl Manual

Creating and Running Custom Tcl Scripts

This topic describes how to easily create Tcl scripts using Diamond's user interface and manual methods. FPGA design using Tcl scripts provides some distinct advantages over using the graphical user interface's lists, views and menu commands. For example, Tcl scripts allow you to do the following:

- Set the tool environment to exactly the same state for every design run. This eliminates human errors caused by forgetting to manually set a critical build parameter from a drop-down menu.
- Manipulate intermediate files automatically, and consistently on every run. For example, EDIF file errors can be corrected prior to performing additional netlist transformation operations.
- Run your script automatically by using job control software. This gives you the flexibilty to run jobs at any time of day or night, taking advantage of idle cycles on your corporate computer system.

Creating Tcl Scripts There are a couple of different methods you can use to create Diamond Tcl scripts. This section will discuss each one and provide step-by-step instructions for you to get started Tcl scripting repetitious Diamond commands or entire workflows.

One method you have available is to use your favorite text editor to enter a sequence of Diamond Tcl commands. The syntax of each the Diamond Tcl commands is available in later topics in this portion of the online help. This method should only be used by very experienced Diamond Tcl command line users.

The preferred method is to let the Diamond GUI assist you in getting the correct syntax for each Tcl command. When you interact with the Diamond user interface each time you launch a *scriptable* process and the corresponding Diamond Tcl command is echoed to the Tcl Console. This makes it much simpler to get the correct command line syntax for each Diamond command. Once you have the fundamental commands executed in the correct order, you can then add additional Tcl code to perform error checking, or customization steps.

To create a Tcl command script in Diamond:

1. Start the Diamond design software and close any project that may be open.
2. In the Tcl Console execute the custom **reset** command. This clears the Tcl Console command history.
3. Use the Diamond graphical user interface to start capturing the basic command sequence. The Tcl Console echos the commands in its window. Start by opening the project for which you wish to create the TCL script.

Then click on the processes in the Process list to run them. For example, run these processes in their chronological order in the design flow:

- Synthesize Design
- Map Design
- Place & Route Design
- Export Files

4. In the Tcl Console window enter the command,

   ```
   save_script <filename.ext>
   ```

   The <filename.ext> is any file identifier that has no spaces and contains no special characters except underscores. For example, **myscript.tcl** or **design_flow_1.tcl** are acceptable save_script values, but **my$script** or **my script** are invalid. The **<filename.ext>** entry can be preceded with a absolute or relative path. Use the **"/"** (i.e. forward slash) character to delimit the path elements. If the path is not specified explicitly the script is saved in the current working directory. The current working directory can be determined by using the TCL *pwd* command.

5. You can now use your favorite text editor to make any changes to the script you feel are necessary. Start your text editor, navigate to the directory the *save_script* command saved the base script, and open the file.

   Note

   In most all cases, you will have to clean up the script you saved and remove any invalid arguments or any commands that cannot be performed in the Diamond environment due to some conflict or exception. You will likely have to revisit this step later if after running your script you experience any run errors due to syntax errors or technology exceptions.

Sample Diamond Tcl Script The following Diamond Tcl script shows a very simple script that opens a project, runs the entire design flow through the Place & Route process, then closes the project. A typical script will contain more tasks and will check for failure conditions. Use this simple example as a general guideline.

Figure 63: Simple Diamond Script

```
prj_project open "C:/lscc/diamond/examples/edif_counter/
edif.ldf"
prj_run PAR -impl edif -forceAll
prj_project close
save_script c:/lscc/diamond/examples/edif_counter/myscript2.tcl
```

Running Tcl Scripts Diamond TCL scripts are run exclusively from the Diamond TCL Console. You can use either the TCL Console integrated into the Diamond UI, or by launching the stand-alone TCL Console.

To run a Tcl script in Diamond:

1. Launch the Diamond GUI, or the stand-alone TCL Console.

Open Diamond but do not open your project. If your project is open, choose **File > Close Project**.

2. If you are using the Diamond main window, click on the **Tcl Console tab** in the Output area at the bottom to open the console.
3. Use the TCL *source* command to load and run your TCL script. The *source* command requires, as it's only argument, the filename of the script you want to load and run. Prefix the script file name with any required relative or absolute path information. To run the example script shown in the previous section use:

   ```
   source C:/lscc/diamond/<version_number>/examples/
   edif_counter/myscript2.tcl
   ```

   As long as there are no syntax errors or invalid arguments, the script will open the project, synthesize, map, and place-and-route the design. Once the design finishes it closes the project. If there are errors in the script, you will see the errors in red in the Tcl Console after you attempt to run it. Go back to your script and correct the errors that prevented the script from running.

See Also ▶"Running the Tcl Console" on page 2587

- "Diamond Tool Tcl Command Syntax" on page 2593
- "Running Tcl Scripts When Launching Diamond" on page 2592
- Tcl Manual

Running Tcl Scripts When Launching Diamond

This topic describes how launch the Diamond software and automatically run Tcl scripts scripts using a command line shell or the stand-alone Tcl console. Your Tcl script can be standard Tcl commands as well as Diamond-specific Tcl commands.

Refer to "Creating and Running Custom Tcl Scripts" on page 2590 for more information on creating custom Tcl scripts.

To launch Diamond and run a Tcl script from a command line shell or the stand-alone Tcl console:

- Enter the following command:

  On Windows:

  ```
  pnmain.exe -t<tcl_path_file>
  ```

  On Linux:

  ```
  diamond -t<tcl_path_file>
  ```

Sample Diamond Tcl Script The following Diamond Tcl script shows a very simple script, running in Windows, that opens a project and runs the design flow through the MAP process. Use this simple example as a general guideline.

Figure 64: Simple Diamond Script

```
prj_project open C:/test/iobasic_diamond/io1.ldf
prj_run Map -impl io1
```

The above example is saved in Windows as the file mytcl.tcl in the directory C:/test. By running the following command from either a DOS shell or the Tcl console in Windows, the Diamond GUI starts, the project i01.ldf opens, and the MAP process automatically runs.

```
pnmain.exe -tc:/test/mytcl.tcl
```

See Also ▶"Running the Tcl Console" on page 2587

- "Diamond Tool Tcl Command Syntax" on page 2593
- "Creating and Running Custom Tcl Scripts" on page 2590
- Tcl Manual

Diamond Tool Tcl Command Syntax

This part of the Tcl Command Reference Guide introduces the syntax of each of the Diamond tools and provides you with examples to help you construct your own commands and scripts.

Diamond tries to make it easy to develop TCL scripts by mirroring the correct command syntax in the Tcl Console based on the actions performed by you in the GUI. This process works well for most designs, but there are times when a greater degree of control is required. More control over the build process is made available through additional command line switches. The additional switches may not be invoked by actions taken by you when using the GUI. This section provides additional information about all of the Tcl commands implemented in Diamond.

The Tcl Commands are broken into major categories. The major categories are:

- Diamond Tcl Console Commands
- Diamond Project Tcl Commands
- Simulation Libraries Compilation Tcl Command
- Reveal Inserter Tcl Commands
- Reveal Analyzer Tcl Commands
- Power Calculator Tcl Commands
- NGD Tcl Commands

- NCD View Tcl Commands
- Programmer Tcl Commands
- Engineering Change Order Tcl Commands
- Incremental Design Flow Tcl Commands

Diamond Tcl Console Commands

The Diamond Tcl Console provides a small number of commands that allow you to perform some basic actions upon the Tcl Console Pane. The Diamond Tcl Console commands differ from the other Tcl commands provided in Diamond. This dtc program's general Tcl Console commands do not use the *dtc_* prefix in the command syntax as is the convention with other tools in Diamond.

Diamond Tcl Console Command Descriptions The following table provides a listing of all valid Diamond Tcl Console-related commands.

Table 915: Diamond Tcl Console Commands

Command	Arguments	Description
history	N/A	The *history* command lists the command history in the Tcl Console that you executed in the current session. Every command entered into the Tcl Console, either by the GUI, or by direct entry in the Tcl Console, is recorded so that it can be recalled at any time. The command history list is cleared when a project is *opened* or when the Tcl Console *reset* command is executed.
reset	N/A	The *reset* command clears anything present in the Tcl Console pane, and erases all entries in the command line history.
clear	N/A	The *clear* command erases anything present in the Tcl Console pane, and prints the current *prompt* character in the upper left corner of the Tcl Console pane without erasing the command history.

Table 915: Diamond Tcl Console Commands

Command	Arguments	Description
save_script	<filename.ext>	Saves the contents of the command line history memory buffer into the script file specified. The script is, by default, stored into the current working directory. File paths using forward slashes used with an identifier are valid if using an absolute file path to an existing script folder.
set_prompt	<new_character>	The default prompt character in the Tcl Console is the “greater than” symbol or angle bracket (i.e., >). You can change this prompt character to some other special character such as a dollar sign ($) or number symbol (#) if you prefer.

Diamond Tcl Console Command Examples This section illustrates and describes a few samples of Diamond Tcl Console commands.

Example 1 To save a script, you simply use the **save_script** command in the Tcl Console window with a name or file path/name argument. In the first example command line, the file path is absolute, that is, it includes the entire path. Here you are saving “myscript.tcl” to the existing current working directory. The second example creates the same “myscript.tcl” file in the current working directory.

```
save_script C:/lscc/diamond/myproject/scripts/myscript.tcl

save_script myscript.tcl
```

See “Creating and Running Custom Tcl Scripts” on page 2590 for details on how to save and run scripts in Diamond.

Example 2 The following **set_prompt** command reassigns the prompt symbol on the command line as a dollar sign ($). The default is an angle bracket or “greater than” sign (>).

```
set_prompt $
```

Example 3 The following **history** command will print all of the command history that was recorded in the current Tcl Console session.

```
history
```

Diamond Project Tcl Commands

The Diamond Project Tcl Commands allow you to control the contents and settings applied to the tools, and source associated with your design. Projects can be opened, closed, and configured to a consistent state using the commands described in this section.

Diamond Project Tcl Command Descriptions The following table provides a listing of all valid Diamond project-related Tcl command options and describes option functionality.

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
prj_project	**new** -name <project name> [-impl <initial implementation name>] [-dev <device name>] [-impl_dir <initial implementation directory>] [-lpf <lpf file name>]	Creates a new project inside the current working directory. The *new* command can only be used when no other project is currently open. The -name <project name> argument specifies the name of the project. This creates a <project name>.ldf file in the current working directory. The -impl <initial implementation name> argument specifies the active implementation when the project is created. If this left unspecified a default implemenation called “Implemenation0” is created. The -dev <device name> argument specifies the FPGA family, density, footprint, performance grade, and temperature grade to generate designs for. Use the Lattice OPN (Ordering Part Number) for the <device name> argument. The -impl_dir <initial implemenation directory> argument defines the directory where temporary files are stored. If this is not specified the current working directory is used. The -lpf <lpf filename> argument specifies the name of the active logical preference file. The <lpf filename> must have a **.lpf** file extension appended to it. The name of the project is used as a default value for the LPF file if this switch is not used.
	close	Exits the current project. Any unsaved changes are discarded.
	open <projectfile.ldf>	Opens the specified project in the software environment.
	save <projectfile.ldf>	Updates the project with all changes made during the current session and the project file is saved.

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
	option [<synthesis | device> [option value]]	Allows you list or modify project options. The *option* function, when used without additional arguments, returns a Tcl list of the options associated with the project. Diamond defines two option argument values: synthesis and device. The synthesis value is the current synthesis tool, which can be either precision or synplify. The device value is the OPN of the current target FPGA. This option can be used to change FPGA device families by specifying a new OPN.
prj_src	**add** [-format VHDL] [-work <VHDL library name>] [-impl <implementation name> [-opt <name=value>]... [-exclude] <source file>...	Adds a VHDL source file to the specified or active implementation. The syntax used for the Add function depends upon the source file's implementation language. [-format VHDL]: This switch forces Diamond to consider the input file to be VHDL regardless of file extension. The switch is not required if the VHDL input files extension is one of: .vhd, or .vhdl. If there is no switch, this command will select a source type based on the file extension name, for example, .v will be Verilog, .vhd will be VHDL , and .rvl will be a Reveal project file. [-work <VHDL lib name>]: Assigns the source code to the specified library name space. [-impl <implementation name>]: This switch is used to add a source file to a Diamond implementation. If this switch is not specified the source file is added to the active implementation. [-opt name=value]: The -opt argument allows you to set a custom, user-defined option. See Example 9 for guidelines and usage. <source file>...: One or more VHDL source files to add to the specified implementation. <source file>...: One or more EDIF source files to add to the specified implementation.

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
prj_src	**add** [-format VERILOG] [-include <include path list>] [-impl <implementation name> [-opt <name=value>]... [-exclude] <source file>...	Adds a Verilog source file to the specified or active implementation. The syntax used for the Add function depends upon the source file's implementation language. [-format VERILOG]: This switch forces Diamond to consider the input file to be Verilog regardless of file extension. The switch is not required if the Verilog input file's extension is .v. If there is no switch, this command will select a source type based on the file extension name, for example, .v will be Verilog, .vhd will be VHDL , and .rvl will be a Reveal project file. [-include <include path list>]: Defines a list of directory search paths for added source. When defining a list of strings, use the semi-colon as a delimiter to seperate the paths. Currently this argument is just used for Verilog source and finds files that use the include keyword. [-impl <implementation name>]: This switch is used to add a source file to a Diamond implementation. If this switch is not specified the source file is added to the active implementation. [-opt name=value]: The -opt argument allows you to set a custom, user-defined option. See Example 9 for guidelines and usage. <source file>...: One or more Verilog source files to add to the specified implementation.

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
prj_src	**add** [-format EDIF] [-impl <implementation name> [-opt name=value]... [-exclude] <source file>...	Adds an EDIF netlist source file to the specified or active implementation. The syntax used for the Add function depends upon the source file's implementation language. [-format EDIF]: This switch forces Diamond to consider the input file to be EDIF regardless of file extension. The switch is not required if the EDIF input files extension is .edi, .edn, .or edf. If there is no switch, this command will select a source type based on the file extension name, for example, .v will be Verilog, .vhd will be VHDL , and .rvl will be a Reveal project file. [-include <include path list>]: Defines a list of directory search paths for added source. When defining a list of strings, use the semi-colon as a delimiter to separate the paths. Currently this argument is just used for Verilog source and finds files that use the include keyword. [-impl <implementation name>]: This switch is used to add a source file to a Diamond implementation. If this switch is not specified the source file is added to the active implementation. [-opt name=value]: The -opt argument allows you to set a custom, user-defined option. See Example 9 for guidelines and usage. <source file>...: One or more EDIF source files to add to the specified implementation.

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
prj_src	**add** [-format SCHEMATIC] [-impl <implementation name> [-opt name=value]... [-exclude] <source file>...	Adds a schematic source file to the specified or active implementation. The syntax used for the Add function depends upon the source files implementation language. [-format Schematic]: This switch forces Diamond to consider the input file to be a Diamond schematic regardless of file extension. If there is no switch, this command will select a source type based on the file extension name, for example, .v will be Verilog, .vhd will be VHDL , and .rvl will be a Reveal project file. [-include <include path list>]: Defines a list of directory search paths for added source. When defining a list of strings, use the semi-colon as a delimiter to separate the paths. Currently this argument is just used for Verilog source and finds files that use the include keyword. [-impl <implementation name>]: This switch is used to add a source file to a Diamond implementation. If this switch is not specified the source file is added to the active implementation. [-opt name=value]: The -opt argument allows you to set a custom, user-defined option. See Example 9 for guidelines and usage. The -exclude argument in prj_src add command will add the source files in a disabled state. <source file>...: One or more Schematic source files to add to the specified implementation.
	enable [-impl <implement name>] <src file> ...	Enables the excluded design sources from the current project, that is, it will activate a source file for synthesis, to be used as a preference, or for Reveal debugging.
	exclude [-impl <implement name>] <src file>	Disables the included design sources from the current project, that is, it will deactivate a source file for synthesis, to be used as a preference, or for Reveal debugging.

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
	remove [-impl <implementation name>] -all | -src <source filename>...	Deletes the specified source files from the specified implementation. If an implementation is not listed explicitly the source files are removed from the active implementation. The source files are not removed from the file system, they are only removed from consideration in the specified implementation.
	option -src <source name> [-impl <implement name>] -src <source name> [-impl <implement name>] -rem <option name>… -src <source name> [-impl <implement name>] [-append] <option name> [option value list]	Allows you to add, list, or remove source options with the name <source name> in the specified or active implementation of the current project. If the -rem option is used, the following option names appearing after it will be removed. If no argument is used (i.e., “prj_src option”), the default is to list all source options. If only the <option name> argument is used (i.e., “prj_project option <option name>), then the value of that option in the project will be returned. The command will set the option value to the option specified by <option name>. If the <option value> is empty then the option will be removed and ignored (e.g., prj_src option -rem).

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
prj_impl	**new** <new impl name> [-dir <implement directory>] [-lpf <lpf file name>] [-strategy <default strategy name>]	Create a new implementation in the current project with '<new impl name>'. A default .lpf file with the same name as the project name will be used if user dose not set the .lpf file name. The new implementation will use the current active implementation's strategy as the default strategy if no valid strategy is set.
	delete <implement name>	Delete the specified implementation in the current project with '< impl name>'.
	option [-impl <implement name>] [-impl <implement name>] -rem <option name>… [-impl <implement name>] [-append] <option name> [option value] -run_flow [<normal|initial|incremental>]	Allows you to add, list, or remove implementation options with the name <implement name> in the specified or active implementation of the current project. If the -rem option is used, the following option names appearing after it will be removed. If no argument is used (i.e., "prj_impl option"), the default is to list all implementation options. If only the <option name> argument is used (i.e., "prj_impl option <option name>), then the value of that option in the project will be returned. The command will set the option value to the option specified by <option name>. If the <option value> is empty then the option will be removed and ignored (e.g., prj_impl option -rem). The -run_flow argument allows you to switch from the normal mode to an "initial" incremental flow mode and "incremental" which is the mode you should be in after an intial design run during the incremental design flow. With no value parameters specified, -run_flow will return the current mode setting.
	active <implement name>	Activates the implementation with the name <implement name>.
	pre_script [-impl <implement name>] <milestone name> <script_file>	List or set the implementation's user script before running. milestone. <milestone name> should be "syn", "map", "par", "export"

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
	post_script [-impl <implement name>] <milestone name> <script_file>	List or set the implementation's user script after running. milestone. <milestone name> should be "syn", "map", "par", "export"
prj_run	<milestone name> [-impl <implement name>] [-task <task name>] [-forceAll|forceOne]	Runs a milestone or task. Milestones refer to processes in the design flow such as synthesis or map and a “task” is a more specific subprocess that is run within that milestone category. Not all milestones have tasks associated with them. The following is a list of all valid milestones with any associated tasks shown in parentheses behind them: Synthesis, Translate, Map (MapTrace, MapVHDLSimFile, MapVerilogSimFile), PAR (PARTrace, IOTiming, SSO), and Export (IBIS, Bitgen, Promgen, TimingSimFileVlg, TimingSimFileVHD). forceOne: reruns only the process specified in <*task name*>. forceAll: reruns all of the process up to <*task name*>.
prj_dev	**set** [-family|-device|-package|-performance|-operation|-part] **set** [-family <family name>] [-device <device name>] [-package <package name>] [-performance <performance grade>] [-operation <operation>] [-part <part name>]	Sets or lists the project’s FPGA device family. For example, you can change the device to specified part, family, device, package or performance grade. Note: The device part used in the project can also be set or listed by using the prj_project option device command.
prj_syn	**set** <synthesis tool>	Sets project default synthesis tool or lists available synthesis tool options. Diamond supports the following synthesis tool options: **synplify** (Synplify Pro), **precision** (Precision), and **lse** (Lattice Synthesis Engine). If you simply enter prj_syn in the command without any parameters, the synthesis tool used in the current project will appear. If no tool has been used, the command will not return any value. Note: The synthesis setting also can be performed by using prj_project option synthesis command.

Table 916: Diamond Project Tcl Commands

Command	Function (Argument)	Description
prj_strgy	**set** [-impl <implementation name>] <strategy name>	Associates the strategy with the specified implementation or the active implementation in the current project.
	set_value [-strategy <strategy name>] <option name1=option value1> <option name2=option value2>...	Sets one or multiple strategy options values.
	list_value [-strategy <strategy name>] <pattern>	Lists value of strategy items.
	list_option <pattern>	Lists strategy option help.
	copy -from <source strategy name> -name <new strategy name> -file <strategy file name>	Copies an existing strategy and saves it to a newly created strategy file.
	new -name <new strategy name> -file <strategy file name>	Creates a new strategy file with the default strategy setting.
	import -name <new strategy name> -file <strategy file name>	Import an existing strategy file.
	delete <strategy name>	Deletes an existing strategy.

Diamond Project Tcl Command Examples This section illustrates and describes a few samples of Diamond Project Tcl commands.

Example 1 To create a new project, your command may appear something like the following which shows the creation of a LatticeSC device.

```
prj_project new -name "m" -impl "m" -dev LFSC3GA15E-5F900C
```

Example 2 To save a project and give it a certain name (save as), use the project save command as shown below:

```
prj_project save "my_project"
```

To simply save the current project just use the save function with no values:

```
prj_project save
```

Example 3 To open an existing project, the command syntax would appear with the absolute file path on your system as shown in the following example:

```
prj_project open "C:/projects/Diamond5/adder/my_project.bdf"
```

Example 4 To add a source file, in this case a source LPF file, use the prj_src add command as shown below and specify the complete file path:

```
prj_src add "C:/my_project/Diamond5/counter/counter.lpf"
```

Example 5 The following example below shows the prj_strgy copy command that uses source strategy “Setting 1” from a certain file path and copies and renames it “my_strategy” in the current project:

```
prj_strgy copy -from "Setting 1" -name "my_strategy"
-file "E:/my_projects/Diamond_prj/ecp2_counter/setting1.sty"
```

Example 6 The following example below shows the prj_strgy list_option command that uses * to show all the option names or you can use a specific one:

```
prj_strgy list_option par*
prj_strgy list_option par_sort_mode
```

Example 7 The following examples below shows the prj_run command being used to run various applications. In this first example, the Place & Route (PAR) tool or “milestone” process implements an output NCD file named “m” and then runs the TRACE task afterward (PARTrace), forcing the flow to only regenerate itself one level:

```
prj_run PAR -impl m -task PARTrace -forceOne
```

Below, Place & Route “milestone” process is run and an NCD file named “cnt_attr” is output to the project folder.

```
prj_run PAR -impl cnt_attr
```

In this final example, synthesis is run resulting in an output netlist file named “cnt_attr” that is generated into your project folder.

```
prj_run Synthesis -impl cnt_attr
```

Example 8 To copy another project strategy that is already established in another Diamond project from your console, use the prj_strgy copy command as shown below and specify the new strategy name and the strategy file name.

```
prj_strgy copy -name new_strategy -file strategy.stg
```

Example 9 The prj_src -opt command allows you to set a custom, user-defined option. This -opt argument value, however, cannot conflict with existing options already in the system, that is, its identifier must differ from system commands such as "include" and "lib" for example. In addition, a user-defined option may not affect the internal flow but can be queried for any usage in a user's script to arrange their design and sources. All user-defined options can be written to the Diamond project LDF file.

In the example below, the -opt argument is used as a qualifier to make a distinction between to .rvl file test cases.

```
prj_src add test1.rvl -opt "debug_case=golden_case"
prj_src add test2.rvl -opt "debug_case=bad_case"
```

Example 10 To set the current synthesis tool in your current project to a valid synthesis tool value, use the following command convention. This command shows how you set “lse” or the Lattice Synthesis Engine. The other values are Synplify and Precision.

```
prj_syn set lse
```

Now type prj_syn and it will return the current value. If you perform this in the Tcl Console, the project summary's current synthesis value will change interactively. If no tool is selected, Diamond will use Synplify by default.

Example 11 After you modify your strategy settings in the Diamond interface the values are saved to the current setting via a Tcl command. For example, a command similar to the following will be called if Synplify frequency and area options are changed.

```
prj_strgy set_value -strategy strategy1 SYN_Frequency=300
SYN_Area=False
```

Simulation Libraries Compilation Tcl Command

This section provides Simulation Libraries Compilation extended Tcl command syntax and usage examples.

Simulation Libraries Compilation Tcl Command Descriptions The following table provides a listing of all valid Simulation Libraries Compilation Tcl Command arguments and describes their usage.

Table 917: Simulation Libraries Compilation Tcl Command

Command	Function (Argument)	Description
cmpl_libs	**-sim_path** <sim_path> [-sim_vendor {mentor<default>}] [-lang {verilog|vhdl|all<default>}] [-device {sc|scm|ec|xp|ecp|machxo|ecp2| ecp2m|xp2|ecp3|machxo2|machxo3 d|machxo3l|lifmd|lifmdf|lptm|lptm2|e cp5u|ecp5um|all<default>}] [-target_path <target_path>]	The -sim_path argument specifies the path to the simulation tool executable (binary) folder. This option is mandatory. Currently only Modelsim and Questa simulators are supported. NOTE: If you are a Windows user and prefer the \ notation in the path, you must surround it with {}. And "" or {} will be needed if the path has spaces. The -sim_vendor argument is optional, and intended for future use. It currently supports only Mentor Graphics simulators (Modelsim / Questa). The -lang argument specifies the HDL type of the compiled libraries. This argument is optional, and the default is to compile both types of libraries. The -device argument specifies the Lattice FPGA device to compile simulation libraries for. This argument is optional, and the default is to compile libraries for all the Lattice FPGA devices. The -target_path argument specifies the target path, where you want the compiled libraries and modelsim.ini file to be located. This argument is optional, and the default target path is the current folder. NOTES: (1) This argument is recommended if you did not change the current folder from the Diamond's startup (binary) folder, or if the current folder is write-protected. (2) If you are a Windows user and prefer the \ notation in the path, you must surround it with {}. And "" or {} will be needed if the path has spaces.

Simulation Libraries Compilation Tcl Command Examples This section illustrates and describes a few examples of Simulation Libraries Compilation Tcl command.

Example 1 The following command will compile all the Lattice FPGA libraries for both Verilog and VHDL simulation, and place them under the folder specified by -target_path. The path to Modelsim is specified by -sim_path.

```
cmpl_libs -sim_path C:/modeltech64_10.0c/win64 -target_path c:/
mti_libs
```

Example 2 The following command will compile the LatticeECP3 libraries for Verilog simulation only, and place them under the current folder. The path to Modelsim is specified by -sim_path.

```
cmpl_libs -sim_path {C:\modeltech_6.6d\win32} -device ecp3 -
lang verilog
```

Reveal Inserter Tcl Commands

This section provides Reveal Inserter extended Tcl command syntax, command options, and usage examples.

Reveal Inserter Tcl Command Descriptions The following table provides a listing of all valid Reveal Inserter Tcl command options and describes option functionality.

Table 918: Reveal Inserter Tcl Commands

Command	Function (Argument)	Description
rvl_core	**set** -name <core name>	Selects core. Needed for all actions except Token. No option, return core names.
	add -name <core name>	Adds a core. Returns added core name.
	delete -name <core name>	Deletes a core. Returns deleted core name.
	rename -name <core name> <new core name>	Renames core. Returns new core name.

Table 918: Reveal Inserter Tcl Commands

Command	Function (Argument)	Description
rvl_trace	**add** -signal <signal list>	Adds signals to trace. Return new signal list. No option, return current signal list.
	delete -signal <signal list>	Delete signals from trace. Return new signal list. No option, return current signal list.
	rename -name<bus name> <new bus name>	Renames existing group. Return new group name.
	set [-sampleclk <clock signal>] [-sampleenable <on | off>] [-bufferdepth <value>] [-implementation <EBR | distram>] [-timestamp <on | off>] [-timestampbits <value>] [-capturemode <single | multiple>] [-capturemin <value>] [-includetrigger <on | off>]	If you specify no set argument options, the command will return a list of trace options. ▶ The -sampleclk argument sets sample clock. ▶ The -sampleenable argument enables or disables the sample enable. ▶ The -bufferdepth argument sets trace buffer depth. ▶ The -implementation argument sets trace implementation to ERB or distributed RAM (distram). ▶ The -timestamp argument turns timestamp on or off. ▶ The -timestampbits arguments sets the timestamp bit width value. ▶ The -capturemode argument sets capture mode. ▶ The -capturemin argument sets the minimum capture value. ▶ The -includetrigger argument sets behavior to include trigger signals as traces.
	set -radix <radix> <signal list>	Sets radix on signals (not implemented yet). Return radix.
	set -group <signal list>	Creates new group with signals.
	set -ungroup <bus list>	Ungroups signals.

Table 918: Reveal Inserter Tcl Commands

Command	Function (Argument)	Description
rvl_tu	**add** -name <TU name>	Adds TU. Return new TU name.
	delete -name <TU name>	Deletes TU. Return deleted name.
	rename -name <name> <new name>	Renames TU. Return new name.
	set [-name <TU name>] [-addsig <signal list>] [-deletesig <signal list>] [-operator {eq | ne | gt | ge| lt | | le | rise | fall | serial] [-value <value> [-radix {bin | oct | dec | hex}]	If you specify no set argument options, the command will return a list of TU. ▶ The -name argument selects the TU for the options. If no TU options, return list of options and value for the selected TU. ▶ The -addsig argument adds signals to TU. ▶ The -deletesig argument deletes signals from TU. ▶ The -operator argument sets comparison operator. Operators are equal to (eq), not equal to (ne), greater than (gt), greater than or equal to (ge), less than (lt), less than or equal to (le), rising edge (rise), falling edge (fall), and serial compare (serial). ▶ The -value argument sets TU value. ▶ The -radix argument sets TU radix. Options are binary (bin), octal (oct), decimal (dec), and hexadecimal (hex).
rvl_te	**add** -name <TE name>	Adds TE. Return new TE name.
	delete -name <TE name>	Deletes TE. Return deleted name.
	rename -name <name> <new TE name>	Renames TE. Return new name.
	set [-name <TE name>] [-expression <expression list>] [-ramtype <EBR/ slices>] [-maxdepth <value>] [-maxcount <value>]	If you specify no set argument options, the command will return a list of TE. ▶ The -name argument selects the TE for the options. If no TE options, return list of options and value for the selected TE. ▶ The -expression argument sets TE expression. ▶ The -ramtype argument sets TE ram type. ▶ The -maxdepth argument sets TE max sequence depth. ▶ The -maxcount argument sets TE max counter value.

Table 918: Reveal Inserter Tcl Commands

Command	Function (Argument)	Description
rvl_trigger	**set** [-defaultradix <radix>] [-finalcounter <on/off>] [-finalcountervalue <value>] [-triggerout <on/off>] [-triggeroutname <name>] [-triggerouttype <IO/NET/Both>] [-triggeroutpolarity <AH/AL>] [-triggeroutmpw <value>]	▶ No options will return all of the trigger options. ▶ The -defaultradix argument sets the default TU radix. ▶ The -finalcounter argument turns the final trigger counter on and off. ▶ The -finalcountervalue argument sets the final trigger counter value. ▶ The -triggerout argument turns the trigger out on and off. ▶ The -triggeroutname argument sets name for trigger out. ▶ The -triggerouttype argument sets type for trigger out. ▶ The -triggeroutpolarity argument sets polarity for trigger out. ▶ The -triggeroutmpw argument sets minimum pulse width for trigger out.
rvl_project	**open** <rvl file>	Opens rvl file.
	save <rvl file>	Saves rvl file.
	new <rvl file>	Creates new rvl file.
	close	Exits program.
	run [-drc] [-debug]	▶ The -drc argument runs DRC. ▶ The -debug argument runs Insert Debug.

Reveal Inserter Tcl Command Examples This section illustrates and describes a few samples of Reveal Inserter Tcl commands.

Example 1 To create a new Reveal Inserter project with the .rvl file extension in your project directory, use the rvl_project command as shown below using the new option.

```
rvl_project new my_project
```

Example 2 The following example shows how to set up TU parameters for Reveal Inserter:

```
rvl_tu set -name TU1 -addsig count[7:0] -operator eq -value C3
-radix hex
```

Reveal Analyzer Tcl Commands

This section provides Reveal Analyzer extended Tcl command syntax, command options, and usage examples.

Reveal Analyzer Tcl Command Descriptions The following table provides a listing of all valid Reveal Analyzer Tcl command options and describes option functionality.

Table 919: Reveal Analyzer Tcl Commands

Command	Function (Argument)	Description
rva_trace	**get**	Lists all trace signals in a core.
rva_core	**set** [-name <name>] [-run <on|off>]	▶ No arguments returns a list of cores. ▶ Selects core. Needed for other actions. ▶ The -run argument urns run check box on/off for core.
rva_tu	**rename** <name> <new name>	This function renames TU.
	set [-name <name>] [-operator {== | != | > | >= | < | <= | "rising edge" | "falling edge"}] [-value <value>] [-radix {bin | oct | dec | hex | <token>}]	▶ No arguments returns a list of TU. ▶ The -name argument selects TU. If no options, returns options and values for the selected TU. ▶ The -operator argument sets the comparison operator. Operators are equal to (==), not equal to (!=), greater than (>), greater than or equal to (>=), less than (<), less than or equal to (<=), "rising edge", "falling edge", and serial compare (serial). ▶ The -value argument sets the TU value. ▶ The -radix argument sets TU radix. Options are binary (bin), octal (oct), decimal (dec), hexadecimal (hex), or the name of a token set.
rva_te	**rename** <name> <new name>	This function renames TE.
	set [-name <name>] [-expression <expression list>] [-enable <on|off>]	▶ No arguments specified will return a list of TE. ▶ The -name argument select TE. If no options are specified, it returns options and values for the selected TE. ▶ The -expression argument sets the TE expression. ▶ The -enable argument enables or disables TE.

Table 919: Reveal Analyzer Tcl Commands

Command	Function (Argument)	Description
rva_tokenmgr	**set_tokenset** <tokenset name> -name <new token set name> -bits <new token bits>	▶ Select specific token set ▶ Rename a token set ▶ Set number of bits in tokens
	add_tokenset [-tokenset <tokenset name>] [-bits <token bits>] [-token <name=value>]	▶ No arguments, add a tokenset with default name and bits ▶ Set token set name ▶ Set token set bits ▶ Add extra tokens
	del_tokenset -all	▶ Delete all token sets
	del_tokenset <token set name>	▶ Delete the specific token set
	set_token <tokenset name> <token name> -name <new token name> -value <new token value>	▶ Select specific token in specific token set ▶ Set token name ▶ Set token value
	add_token <token set name> <name=value>	▶ Add a token with new name and value in a specific token set
	del_token <token set name> <token name>	▶ Delete a specific token in a specific token set
	export <file name>	▶ Export all token sets to a specific file
	import <file name>	▶ Import and merge all token sets from a specific file
rva_trigoptn	**set** [-teall <AND|OR>] [-finalcounter <on|off>] [-finalcountervalue <value>] [-samples <value>] [-numtriggers <value>] [-position <pre|center|post|value>]	▶ No arguments specified will return list of options. ▶ The -teall argument sets AND ALL or OR ALL for all TEs. ▶ The -finalcounter argument turns final trigger counter on/off. ▶ The -finalcountervalue argument sets final trigger counter value. ▶ The -samples argument sets number of samples to capture. ▶ The -numtriggers argument sets number of triggers to capture. ▶ The -position argument sets trigger position to pre-selected or user value.

Table 919: Reveal Analyzer Tcl Commands

Command	Function (Argument)	Description
rva_project	**open** <rva file>	The open function with a rva file argument opens an rva file.
	save <rva file>	Saves rva file.
	new [-rva <rva file>] [-xcf <xcf file>] [-rvl <rvl file>] [-dev <device:id>] -port <number> -cable <type>	Create new rva file with required options. An XCF file is required for multiple devices. Cable type values are LATTICE|USB|USB2 but optional for single devices on cable.
	close	Exits program.
	export -vcd <file name> [-module <title>]	Export VCD file. Optional to include a title in the VCD file. If the -module argument is not included, the title will be “<unknown>”.
	export -txt <file name> [-siglist <signal list>]	Export TEXT file; optional to export selected signal list only, by default all signals are exported.
	set [-frequency <frequency value>] [-period <period value>] [-tckdelay <delay value>] [-cabletype <type>] [-cableport <number>]	▶ No arguments specified will return options. ▶ The -frequency argument sets the frequency value for sample clock in MHz. ▶ The - period argument sets a period value for sample clock in ns or ps. ▶ The -tckdelay argument sets a TCK clock pin pulse width delay value (MachXO2) ▶ The -cabletype argument changes the type of cable. Values are LATTICE|USB|USB2. ▶ The -cable port argument changes the port.
	run	Runs until trigger condition to capture data.
	stop	Stops without capturing data.
	manualtrig	Manual Trigger to capture data.
	run -download [-xcf <xcf file>] [-log <log file>]	Downloads program using ispVM.

Reveal Analyzer Tcl Command Examples This section illustrates and describes a few samples of Reveal Analyzer Tcl commands.

Example 1 To create a new Reveal Analyzer project with the .rvl file extension in your project directory, use the rva_project command as shown below using the new option.

```
rva_project new my_project
```

Example 2 The following command line example shows how to specify a new project that uses a parallel cable port.

```
rva_project new –rva untitled –rvl "count.rvl" –dev "LFXP2-
5E:0x01299043" –port 888 –cable LATTICE
```

Example 3 The following example shows how to set up TU parameters for Reveal Analyzer:

```
rva_tu set -name TU1 -operator eq -value 10110100 -radix bin
```

Example 4 In this example, a MachXO2 Pico demo board requires a setting on TCK Low Pulse Width Delay to equal 2.

```
rva_project set –tckdelay 2
```

Power Calculator Tcl Commands

This section provides Power Calculator extended Tcl command syntax, command options, and usage examples.

Power Calculator Tcl Command Descriptions The following table provides a listing of all valid Power Calculator Tcl command options and describes option functionality.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_command	**open** -project <project file>	Opens project file.
	open -vcd <vcd file>	Opens vcd file.
	save -project <project file>	Saves project file.
	new -project <project file>	Creates new project.
	exit	Exits program.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_device	**list**	Lists all device info.
	list -family	Lists all current device families.
	list -device	Lists all current devices.
	list -packagetype	Lists all current package types.
	list -performance	Lists all valid device performance grades.
	list -operatingconditions	Lists all operating conditions settings.
	list -partnames	Lists all device part names.
	set -family <value>	Sets device family.
	set -device <value>	Sets device.
	set -packagetype <value>	Sets package type.
	set -performance <value>	Sets performance grade.
	set -operatingconditions <value>	Sets operating condition.
	set -partnames <value>	Sets part name.
pwc_parameters	**set** -process <value>	Sets device power process type.
pwc_thermal	**list**	Lists all thermal settings.
	set -userdefined <theta-ja value>	Sets user defined theta JA.
	set -board <value>	Sets board type.
	set -heatsink <value>	Sets heatsink type.
	set -airflow <value>	Sets airflow value.
	set -ambient <value>	Sets ambient temperature value.
pwc_settings	**list**	Lists all Power Calculator settings.
	set -af <value>	Sets default activity factor.
	set -freq <value>	Sets default frequency.
	set -freqtwr <file>	Sets usage twr file.
	set -freqtwropt <min | pref | trace>	Sets twr file options.
	set -estrouting <low | medium | high>	Sets estimated routing option.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_supply	**list**	Lists all power supplies.
	set -type <value> -mult <value> -voltage <value>	Sets multiplication factor and voltage of named power supply.
	set -type <value> -mult [<value>]	Sets multiplication factor of named power supply.
	set -type <value> -voltage [<value>]	Sets multiplication factor of named power supply.
pwc_logicblocks	**list**	Lists all logic blocks.
	add [-clockname <value>] [-freq <value>] [-af <value>] [-logicluts <value>] [-distram <value>] [-rippleslices <value>] [-registers <value>]	Adds logic block row.
	remove [-clockname <value>] [-freq <value>] [-af <value>] [-logicluts <value>] [-distram <value>] [rippleslices (value)] [-registers <value>]	Removes a logic block row.
pwc_clocks	**list**	Lists all clocks.
	add [-clockname <value>] [-freq <value>]	Adds clock.
	remove [-clockname <value>] [-freq <value>]	Removes clock.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_inout	**list** -io	List all I/Os.
	add -io [-clockname <value>] [-type <value>] [-ioregistermode <value>] [-freq <value>] [-af <value>] [-ip <value>] [-op <value>] [-cload <value>] [-bank <value>] [-pg <value>]	Adds I/O. The -pg argument is for MachXO2 only.
	remove -io [-clockname <value>] [-type <value>] [-ioregistermode <value>] [-freq <value>] [-af <value>] [-ip <value>] [-op <value>] [-cload <value>] [-bank <value>] [-pg <value>]	Removes I/O. The -pg argument is for MachXO2 only.
	list -bidir	Lists all bi-directional I/Os.
	add -bidir [-clockinputname <value>] [-type <value>] [-ioregistermode <value>] [-inputfreq <value>] [-inputaf <value>] [-bidi <value>] [-clockoutputname <value>] [-outputfreq <value>] [-outputaf <value>] [-dutycycle <value>] [-cload <value>] [-bank <value>] [-pg <value>]	Adds bi-directional I/O. The -pg argument is for MachXO2 only.
	remove -bidir [-clockinputname <value>] [-type <value>] [-ioregistermode <value>] [-inputfreq <value>] [-inputaf <value>] [-bidi <value>] [-clockoutputname <value>] [-outputfreq <value>] [-outputaf <value>] [-dutycycle <value>] [-cload <value>] [-bank <value>] [-pg <value>]	Removes bi-directional I/O. The -pg argument is for MachXO2 only.
	list -bank	Lists all bank voltages.
	set -bank <bank id> -inrdshutoff | lvdsoshutoff [<value>]	(MachXO2 only) Sets InRD Shut-off and/or LVDSO Shut-off for bank. XO2 only. The bank ID is a string and refers to the bank number. The value set for Allow InRD Shut-off or Allow LVDSO Shut-off can be “yes” or “no”.
	list -term	Lists all termination resistances and voltages.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_blockram	**list** -spram	Lists all single port block RAMs.
	add -spram [-clockname <value>] [-ebrblocks <value>] [-freq <value>] [-af <value>]	Adds single port block RAMs.
	remove -spram [-clockname <value>] [-ebrblocks <value>] [-freq <value>] [-af <value>]	Removes single port block RAMs.
	list -dpram	Lists all dual port block RAMs.
	add -dpram [-rdclockname <name>] [-rdclkfreq <value>] [-rdaf <value>] [-ebrblocks <value>] [-wrclockname <name>] [-wrclkfreq <value>] [-wraf (<value>]	Adds dual port block RAMs.
	remove -dpram [-rdclockname <name>] [-rdclkfreq <value>] [-rdaf <value>] [-ebrblocks <value>] [-wrclockname <name>] [-wrclkfreq <value>] [-wraf (<value>]	Removes all dual port block RAMs.
	list -dpramtrue	Lists all dual port true block RAMs.
	add -dpramtrue [-clockaname <name>] [-clkafreq <value>] [-clkaaf <value>] [-ebrblocks <value>] [-clockbname <value>] [-clkbfreq <value>] [-clkbaf <value>]	Adds dual port true block RAMs.
	remove -dpramtrue [-clockaname <name>] [-clkafreq <value>] [-clkaaf <value>] [-ebrblocks <value>] [-clockbname <value>] [-clkbfreq <value>] [-clkbaf <value>]	Removes dual port true block RAMs.
	list -fifodc	Lists all FIFO DC block RAMs.
	add -fifodc [-rdclockname <name>] [-outputregistersrd <value>] [-rdclkfreq <value>] [-rdaf <value>] [-ebrblocks <value>] [-wrclockname <name>] [-outputregisterswr <value>] [-wrclkfreq <value> [-wraf <value>]	Adds FIFO DC block RAMs.
	remove -fifodc [-rdclockname <name>] [-outputregistersrd <value>] [-rdclkfreq <value>] [-rdaf <value>] [-ebrblocks <value>] [-wrclockname <name>] [-outputregisterswr <value>] [-wrclkfreq <value> [-wraf <value>]	Removes FIFO DC block RAMs.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_dspblock	**list**	Lists all DSP blocks.
	add [-clockname <name>] [-freq <value>] [-af <value>] [-type <value>] [-dsp <vaue>]	Adds DSP blocks.
	remove [-clockname <name>] [-freq <value>] [-af <value>] [-type <value>] [-dsp <value>]	Removes DSP blocks.
pwc_plldll	**list** -pll	Lists all PLLs.
	add -pll [-outputclockname <name>] [-inputfreq <value>] [-n <value>] [-v <value>] [-m <value>] [-pll <value>]	Adds PLL.
	remove -pll [-outputclockname <name>] [-inputfreq <value>] [-n <value>] [-v <value>] [-m <value>] [-pll <value>]	Removes PLL.
	list -dll	Lists all DLLs.
	add -dll [-clockname <value>] [-freq (<value>] [-dll <value>]	Adds DLLs.
	remove -dll [-clockname <value>] [-freq (<value>] [-dll <value>]	Removes DLLs.
	list -dqsdll	Lists all DQSDLLs.
	add -dqsdll [-clockname <value>] [-freq (<value>] [-dqsdll <value>]	Adds DQSDLLs.
	remove -dqsdll [-clockname <value>] [-freq (<value>] [-dqsdll <value>]	Removes DQSDLLs.
	list -dqs	(MachXO only) Lists all DQS.
	add -dqs [-clockname <value>] [-freq (<value>] [-dqs <value>]	(MachXO only) Adds DQS.
	remove -dqs [-clockname <value>] [-freq (<value>] [-dqs <value>]	(MachXO only) Removes DQS.
	list -dlldel	(MachXO only) Lists all DQS.
	add -dlldel [-clockname <value>] [-freq (<value>] [-dlldel <value>]	(MachXO only) Adds DQS.
	remove -dlldel [-clockname <value>] [-freq (<value>] [-dlldel <value>]	(MachXO only) Removes DLLDEL.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_maco	**list**	Lists all MACOs.
	add [-clockname <name>] [-freq <value>] [-af <value>] [-type <value>] [-maco <value>]	Adds MACOs.
	remove [-clockname <name>] [-freq <value>] [-af <value>] [-type <value>] [-maco <value>]	Removes MACOs.
pwc_serdes	**list**	Lists all MACOs.
	add [-clockname <name>] [-freq <value>] [-channels <value>] [-gearingratio <value>] [-txpreemphasis <value>]	Adds SERDES for LatticeSC/SCM devices.
	add [-clockname <name>] [-freq <value>] [-channels <value>] [-mode <value>]	Adds SERDES for LatticeECP3/ ECP2M/ECP2MS devices.
	remove [-clockname <name>] [-freq <value>] [-channels <value>] [-gearingratio <value>] [-txpreemphasis <value>]	Removes SERDES for LatticeSC/SCM devices.
	remove [-clockname <name>] [-freq <value>] [-channels <value>] [-mode <value>]	Removes SERDES for LatticeECP3/ ECP2M/ECP2MS devices.
pwc_writereport	txt -file <file name>	Generates text report and writes to file.
	html -file <file name>	Generates HTML report and writes to file.
pwc_efb	**list** [-i2c1 | -i2c2 | -spi | -ufm | -tc | -wishbone]	(MachXO2 only) Outputs a value list of all rows in the given EFB table. Tables include I2C1, I2C2, SPI, UMF, TC, and WISHBONE, respectively. If you do not provide an EFB value, a list of all the rows for each type will be output.
	add [-i2c1 <key value>] | [-i2c2 <key value>] | [-spi <key value>] | [[-ufm <key value>] | [-tc <key value>] | [-wishbone <key value>]	(MachXO2 only) Adds rows to a given EFB table. Key values are rows of the I2C1, I2C2, SPI, UMF, TC, and WISHBONE tables, respectively. Outputs a value list of added rows in the given table.
	remove [-i2c1 <key value>] | [-i2c2 <key value>] | [-spi <key value>] | [[-ufm <key value>] | [-tc <key value>] | [-wishbone <key value>]	(MachXO2 only) Removes rows from a given EFB table. Key values are rows of the I2C1, I2C2, SPI, UMF, TC, and WISHBONE tables, respectively. Outputs a value list of the number of removed rows in the given table.

Table 920: Power Calculator Tcl Commands

Command	Function (Argument)	Description
pwc_misc	**list** [-mclk | -por | -bandgap]	(MachXO2 only) Outputs a value list of all rows in the given EFB table. Tables include MCLK, POR, and BANDGAP, respectively. If you do not provide an EFB value, a list of all the rows for each type will be output.
	set -mclk [-allowstandby <value>] [-mclkfreq <mclk freq>] [-sedfreq <sedfreq>] | -por -allowstandby [<value>] | -bandgap -allowstandby [<value>]	(MachXO2 only) The -mclk command sets standby, MCLK frequency and SED frequency of MCLK. The -allowstandby argument takes No or Yes values. Frequency values are an integer value. This command outputs a list of values in an MCLK table. The -por and -bandgap commands set standby for POR and BANDGAP in similar fashion, respectively.
pwc_power	**set** -standby [<-bandgap | -por | -osc | -pll <pll id> > Disable|Enable] dynamic [-bank <bank id> [-inrd|-lvdso|-pg Disable|Enable]]	(MachXO2 only) Sets standby and/or bank dynamic option. The PLL ID can be 0 or 1. The bank ID is the name of the bank. A value list of standby options and/or dynamic bank options are output. In addition, the Disable or Enable standby option is output.

Power Calculator Tcl Command Examples This section illustrates and describes a few samples of Power Calculator Tcl commands.

Example 1 To set the usage of the .twr file to “ON” in Power Calculator as shown in the following command below:

```
pwc_settings -freqtwr on
```

Example 2 The follow command below creates a PWC project (.pcf) file namced “abc.pcf” from an input NCD file named “abc.ncd”:

```
pwc_command new -project abc.pcf -ncd abc.ncd
```

Example 3 To add a row in the logic table with 4 logic LUTs and 5 registers (other column value is default), use a command as follows:

```
pwc_logicblocks add –logicluts 4 –registers
```

This example adds a row in the logic table with default values:

```
pwc_logicblocks add5
```

Note

You can see that the keywords “logicluts” and “registers” are just the same as the column name in the table. The keyword, however, is in lower case and is an alphanumeric character, and by rule does not include blank spaces and characters in paretheses. This rule is also applied in other component tables.

Example 4 To see the current device family use the first command shown below. To change the current device family setting to MachXO, for example, use the pwc_device set" command with the -family option:

```
pwc_device set –family
pwc_device set –family MachXO
```

Example 5 To set the default frequency to, for example, 100 Mhz:

```
pwc_settings set –freq 100
```

Example 6 To add one I2C1 in an EFB panel for a MachXO2 device, use a similar command convention to the one shown below:

```
pwc_efb add –i2c1 –i2c1 1
```

Note

The first i2c1 is the table name and the second is column name.

Example 7 The command below saves the current project to a new name:

```
pwc_command save –project newname.pcf
```

Example 8 To create an HTML report, you would run a command like the one shown below:

```
pwc_writereport  html –file c:/abc.html
```

NGD Tcl Commands

This section provides NGD extended Tcl command syntax, command options, and usage examples.

NGD Tcl Command Descriptions The following table provides a listing of all valid NGD Tcl command options and describes option functionality.

Table 921: NGD Tcl Commands

Command	Function (Argument)	Description
ngd_port	**get** [-in] [-out] [-bidi] [-tristate]	Returns list of port tcl objects in a comma-separated string in following format: port_name1,port_name2,port_name3 An empty string is returned if there is no port. If no option is specified then all ports will be returned. If more than one option is specified then a combination of results from each option is returned.
ngd_inst	**get** [-asic] [-cell] [-comp]	Returns a list of instance tcl objects in a comma-separated string in following format: hier_inst_name1,hier_inst_name2,hier_inst_name3 An empty string is returned if there is no instance. If no option is specified then all instances will be returned. If more than one option is specified then the combination of results from each option is returned.
ngd_net	**get** [-clock] [-vcc] [-gnd]	Returns a list of net/signal tcl objects in a comma-separated string in following format: hier_net_name1,hier_net_name2,hier_net_name3 An empty string is returned if there is no net. If no option is specified then all nets will be returned. If more than one option is specified then the union of results from each option is returned.
	get -inst hierInstName [-pin index | -fanin | -fanout]	Returns a list of net/signal tcl objects in a comma-separated string in following format: hier_net_name1,hier_net_name2,hier_net_name3 An empty string is returned if there is no net.

Table 921: NGD Tcl Commands

Command	Function (Argument)	Description
ngd_attr	**get** -inst hierInstName [attributeName] Acceptable attributeName values are as follows: -pincount -fanincount -fanoutcount -type: possible returned values are "asic", "cell", or "comp"	Returns a list of attributes for specified hierarchical instances.
	get -net netName [attributeName] Acceptable attributeName values are as follows: -fanincount: -fanoutcount: -fun: clock, data value: possible returned values are “gnd”, “vcc”, or “signal”	Returns a list of attributes for specified nets.
	get -port portName [attributeName] Acceptable attributeName values are as follows: -differential: possible returned values are “pos”, “neg”, or “none” - direction: possible returned values are “in”, “out”, or “bidi”	Returns a list of attributes for specified ports.

NGD Tcl Command Examples This section illustrates and describes a few samples of NGD Tcl commands.

Example 1 To return a list of port Tcl objects in your console, use the ngd_port command as shown below and specify the port type. This example uses the bidirectional port (-bidi) port type parameter.

```
ngd_port get -bidi
```

NCD View Tcl Commands

This section provides NCD view extended Tcl command syntax, command options, and usage examples.

NCD View Tcl Command Descriptions The following table provides a listing of all valid NCD view Tcl command options and describes option functionality.

Table 922: NCD View Tcl Commands

Command	Function (Argument)	Description
ncd_port	**get** [-in] [-out] [-bidi]	Returns list of port tcl objects in a comma-separated string in following format: port_name1,port_name2,port_name3 An empty string is returned if there is no port. If no option is specified then all ports will be returned. If more than one option is specified then a combination of results from each option is returned.
	export [{-csv | -lpf} filename [filePath]]	Exports pin configuration to user-specified file in either CSV or LPF format. CSV formation is generated by exporting Device Pin Layout and LPF format is generated through NCD back-annotation. If a filePath is not specified then ./ or current directory is the default.
ncd_inst	**get** [-slice] [-dcs] [-ebr] [-dsp] [-pio]	Returns list of hierarchical instance tcl objects in a comma-separated string in following format: hier_inst_name1,hier_inst_name2,hier_inst_name3 An empty string is returned if there is no port. If no option is specified then all ports will be returned. If more than one option is specified then a combination of results from each option is returned.
ncd_net	**get** -inst hierInstName [-pin | -fanin | -fanout]	For the “pin” option, enter values from 0 to max number of nets of that instance. Returns a list of net/signal tcl objects in a comma-separated string in the following format: hier_net_name1,hier_net_name2,hier_net_name3 An empty string is returned if there is no net.

Table 922: NCD View Tcl Commands

Command	Function (Argument)	Description
ncd_attr	**get** -inst hierInstName [attributeName] Acceptable attributeName values are as follows: pincount: fanincount: fanoutcount: type: slice, dcs, ebr, dsp, pio placed: yes or no	Returns a list of attributes for specified hierarchical instances.
	get -net netName [attributeName] Acceptable attributeName values are as follows: fanincount: fanoutcount: routed: yes, no or partial	Returns a list of attributes for specified nets.
	get -port portName [attributeName] Acceptable attributeName values are as follows: direction: in, out, bidi configuration: List of supported configurations. PCICLAMP, IO_TYPE, PULLMODE, SLEWRATE, and OPENDRAIN are device dependent. Please refer to Spreadsheet View topic in help for more information.	Returns a list of attributes for specified ports.

NCD View Tcl Command Examples This section illustrates and describes a few samples of NCD Tcl commands.

Example 1 To return a list of port Tcl objects in your console, use the ncd_port command as shown below and specify the port type. This example uses the bidirectional port (-bidi) port type parameter.

```
ncd_port get -bidi
```

Example 2 To return a list of net Tcl objects in your console, use the ncd_net command as shown below.

```
ncd_net get -inst hierInstName1
```

Programmer Tcl Commands

This section provides the Programmer extended Tcl command syntax, command options, and usage examples.

Programmer Tcl Command Descriptions The following table provides a listing of all valid Programmer Tcl command options and describes option functionality.

Table 923: Programmer Tcl Commands

Command	Function (Argument)	Description
pgr_project	**open** <project_file>	The open command will open the specified project file in-memory.
	save [<file_path>]	Writes the current project to the specified path. If there is no file path specified then it will overwrite the original file.
	close	Closes the current project. If a Programmer GUI is open with the associated project, then the corresponding Programmer GUI will be closed as well.
	check	Runs the check command on the current project.
	help	Displays help for the pgr_project command.

Table 923: Programmer Tcl Commands

Command	Function (Argument)	Description
pgr_program	<no_argument>	When pgr_program is run without arguments it will display the current status of the available settings. Note that specifying a key without a value will display the current value. The following keys can be used to modify those settings. Generally, the pgr_program command and its sub-commands allow you to run the equivalent process commands from the TCL Console window in the Diamond interface. These commands can override connection options that are set in user defaults.
	cable <lattice | vantis | usb>	Sets the cable type.
	portaddress <0x0378 | 0x0278 | 0x03bc | 0x0378 | 0x0278 | 0x03bc | 0x<custom address> | Ezusb-0 | Ezusb-1 | Ezusb-2 | Ezusb-3 | Ezusb-4 | Ezusb-5 | Ezusb-6 | Ezusb-7 | Ezusb-8 | Ezusb-9 | Ezusb-10 | Ezusb-11 | Ezusb-12 | Ezusb-13 | Ezusb-14 | Ezusb-15>	Sets the port address for the corresponding cable type.
	run [-progmode <turbo | sequential>] [cable <lattice | vantis | usb>] [portaddress <0x0378 | 0x0278 | 0x03bc | 0x0378 | 0x0278 | 0x03bc | 0x<custom address> | Ezusb-0 | Ezusb-1 | Ezusb-2 | Ezusb-3 | Ezusb-4 | Ezusb-5 | Ezusb-6 | Ezusb-7 | Ezusb-8 | Ezusb-9 | Ezusb-10 | Ezusb-11 | Ezusb-12 | Ezusb-13 | Ezusb-14 | Ezusb-15>]	Executes current project and project settings on the hardware. Note that there may be warnings that are displayed in the TCL Console window. These warnings will be ignored and processing will continue.
	help	Displays help for pgr_program command.
pgr_genfile	<no_argument>	The pgr_genfile command allows you to generate data files based on the currently loaded Programmer project and specify some commonly used options.
	process <svf | vme12 | slim>	Sets file type for file generation.
	slimalgorithm <file path>	Sets algorithm file for Slim VME file.
	outfile <file path>	Sets output file for pgr_genfile run.
	run	Generates file based on current settings and current project.
	help	Displays help for pgr_genfile command.

Programmer Tcl Command Examples This section illustrates and describes a few samples of Programmer Tcl commands.

Example 1 The first command below opens up a Programmer XCF project file that exists in an absolute path on the system. There can be many programming files associated with one project. In the GUI interface, the boldfaced file in Diamond is the active project file. For command line, you need to know what the active file is? <need to know this>

```
pgr_project /home/mdm/config_file/myfile.xcf
```

The second command in this example runs the Check action, and then executes all of the operations that have been set up for each device. If Check returns unsuccessful, then the operations will not be executed unless your configuration settings allow for downloading on errors. In the GUI settings you can enable the “Continue Download on Error” option in Programming Settings. <is there an option for this via command line to download on error?>

```
pgr_project check
```

Example 2 The following command runs programming in turbo mode using a USB cable at port address “Ezusb-7”.

```
pgr_program run progmode turbo cable usb portaddress Ezusb-7
```

Example 3 The following command sets the file generation type for JTAG SVF file and generates an output file to “mygenfile.xxx” in a relative path.

```
pgr_genfile run process svf outfile ../genfiles/mygenfile.xxx
```

Engineering Change Order Tcl Commands

This section provides Engineering Change Order (ECO) extended Tcl command syntax, command options, and usage examples available in Diamond Software Tcl (pnmainc.exe).

ECO Tcl Command Descriptions The following table provides a listing of all valid ECO Tcl command options and describes option functionality.

Table 924: ECO Tcl Commands

Command	Function (Argument)	Description
eco_design	**open** -ncd <ncd file> | [-prf <prf file>] [-macro]	Opens an existing design or macro.
	save [-ncd <ncd file> -prf <prf file>] | [-nmc <nmc file>]	Saves the design or macro that is currently open or saves it in a different location or with a different .ncd file name.
	close	Closes the design or macro that is currently open.

Table 924: ECO Tcl Commands (Continued)

Command	Function (Argument)	Description
eco_config	**sysio** -comp <comp name> [-bank_vccio <bank_vccio>] [-clamp <clamp>] [-diffcurrent <diffcurrent>] [-diffdrive <diffdrive>] [-diffresistor <diffresistor>] [-drive <drive>] [-eq_call <eq_call>] [-hysteresis <hysteresis>] [-impedance <impedance>] [-inf <inf>] [-io_type <io_type>] [-multdrive <multdrive>] [-opendrain <opendrain>] [-pciclamp <pciclamp>] [-pullmode <pullmode>] [-pwrsave <pwrsave>] [-refcircuit <refcircuit>] [-slewrate <slewrate>] [-terminategnd <terminategnd>] [-terminatevccio <terminatevccio>] [-terminatevtt <terminatevtt>] [-termination <termination>] [-vcmt <vcmt>] [-vref <vref>]	Edit sysIO preferences after the design has been placed and routed
	pll -comp <comp name> -type <type> [-phaseadj <phaseadj>] [-delay_val <delay>] [-fdel <fdel>] [-clkos_fdel <clkos_fdel>]	Edit sysIO preferences after the design has been placed and routed
	memufm -comp <comp name> -init_all_0 <no|yes> [-format <hex|bin>] [-mem <mem file>]	Update initial values for User Flash Memory
	memebr -instance <instance name> -init_all <0|1|no> -mem <file name> -format <hex|bin|add> -init_data <dynamic|static> -module <module name> -mode <mode value> -depth <depth value> -widtha <widthA value> -widthb <widthB value> [-area <area value>] [-ecc <ecc value>] [-bytesize <bytesize value>]	Update initial values for EBRs. Note: <mode value> is case-insensitive. It only accepts "ROM", "RAM_DQ", "RAMDPS", "RAM_DP", & "RAM_DP_TRUE".
	memdram -instance <instance name> -mem <file name> -format <hex|bin|add> -module <module name> -depth <depth value> -widtha <widthA value>	Update initial values for DRAMs
	prim -comp <comp name> -prim <mode value pair> [-prop <prop value pair>]... [-pin <pin value pair>]...	Update component's primitive settings
eco_add	**sigprobe** -site <site name> -net <net name>	Add signal probes
eco_delete	**sigprobe** -site <site name> -net <net name>	Remove signal probes

ECO Tcl Command Examples This section illustrates and describes a few samples of ECO Tcl commands.

Example 1 The following demonstrates the `sysio` command:

```
eco_config sysio -comp {c[0]} -bank_vccio 3.3 -pullmode UP
```

Example 2 The following demonstrates the `pll` command:

```
eco_config pll -comp {PLLInst_0} -type EHXPLLF -phaseadj 22.5 -
delay_val 2
```

Example 3 The following demonstrates the `memufm` command:

```
eco_config memufm -comp EFBInst_0 -init_all_0 no -format bin -
mem {d:/demo/mem.mem}
```

Example 4 The following demonstrates the `memufm` command:

```
eco_config memufm -comp {EFBInst_0} -init_all_0  yes
```

Example 5 The following demonstrates the `memebr` command:

```
eco_config memebr -instance .ip_rom -init_all no -mem {D:/UT/
Diamond/eco/Memory Initialization/ip_rom/test.mem} -format bin
-init_data dynamic -module ip_rom -mode ROM -depth 256 -widtha
8 -widthb 8
```

Example 6 The following demonstrates the `memdram` command:

```
eco_config memdram -instance u1 -format bin -mem {D:/UT/
Diamond/eco/Memory Initialization/mem_init/mem_init/mem.mem} -
module dram -depth 32 -widtha 16
```

Example 7 The following demonstrates the `memprim` command:

```
eco_config prim -comp master_dcua0_block -prim {DCUA} -prop
{D_CDR_LOL_SET=0b01} -prop {D_PLL_LOL_SET=0b01}
```

Example 8 The following demonstrates the `eco_add` command:

```
eco_add sigprobe -site A2 -net {clk_c}
```

Example 9 The following demonstrates the `eco_delete` command:

```
eco_delete sigprobe -site A2 -net {clk_c}
```

Example 10 The following demonstrates updating memory file in batch using ECO Tcl `eco_design` commands:

1. Save the following commands into a Tcl script file named “meminit.tcl”.

   ```
   eco_design open -ncd {C:/case/trilby_test_impl1.ncd} -prf
   {C:/case/trilby_test_impl1.lpf}
   eco_config memebr -instance {I492/rom4kx8_inst} -init_all no
   -mem {C:/case/trilby_8051.mem} -format add -init_data static
   -module {rom4kx8} -mode {RAM_DP_TRUE} -depth {4096} -widtha
   {8} -widthb {8}
   eco_design save -ncd {C:/case/trilby_test_impl1.ncd}
   eco_design close
   ```

2. From the command line, change the current path to the project path, and run the batch file.
   - On Windows, from the command line type:

     ```
     c:\lscc\diamond\<version_number>\bin\nt\fpgac meminit.tcl
     ```

   - On Linux, from the command line type:

     ```
     cd /usr/local/diamond/<version_number>/bin/lin
     source setupenv
     cd <project path>
     /usr/local/diamond/<version_number>/bin/lin/fpgac
     meminit.tcl
     ```

See Also "Launching the EPIC Tcl Console" on page 2588

Incremental Design Flow Tcl Commands

This section provides Incremental Design Flow Tcl command syntax, command options, and usage.

Incremental Design Flow Tcl Command Descriptions The following table provides a listing of all valid Incremental Design Flow command options and describes option functionality.

Table 925: Incremental Design Flow Tcl Commands

Command	Function (Argument)	Description
prj_incr	**set** [-enable | -disable] [-impl]	Get or set the incremental design flow mode. If you enter prj_incr without options, this will check the current implementation mode (normal or incremental). If you include the options (-enable or -disable) this will enable the incremental flow for the active implementation or for the specified implementation.
	restore [-impl] [-previous | -golden]	Set the previous or golden backup as the reference for the next incremental run.
	backup_golden [-impl]	Set the current successful results as the golden reference backup.
icf_data	**reload**	Loads or reloads the settings of the partitions from the data on disk. The changes in memory will be discarded.
	save	Saves the partition setting changes to the data on disk.

Table 925: Incremental Design Flow Tcl Commands

Command	Function (Argument)	Description
icf_part	**set_level** -part <partition name> [-value <SYNTHESIS | MAPPED | PLACED | ROUTED>]	Sets the preservation level for a partition.
	set_effort -part <part name> [-value <GUIDED | UNGUIDED>]	Sets the reimplementation effort for a partition.
	set_bbox -part <partition name> [-width <value>] [-height <value>]	Sets the bounding box for a partition.
	set_anchor -part <partition name> [-anchor <RxCy>]	Sets the anchor for a partition.
	set_color -part <partition name> [-color <RxxxGxxxBxxx>]	Sets the display color of the partition in the graphical user interface, The “xxx” represents a three-digit number ranging from 000 to 255 for the red, green, and blue value.

Note

The **idf_database** and **idf_partition** commands used in previous versions of Lattice Diamond have been removed, Starting with Lattice Diamond v2.1, these commands are replaced with the **ifc_data** and **ifc_part** commands described above.

Incremental Design Flow Tcl Command Examples The following are a few examples of tcl command lines and a description of what each does for Incremental Design.

Example 1 The following example shows the Tcl command usage for loading the partition information.

```
icf_data reload
```

Example 2 The following example shows the Tcl command usage for saving the partition information.

```
icf_data save
```

Example 3 The following example illustrates the Tcl command usage for setting the preservation level for a partition.

```
icf_part set_level -part CGROUP_GRAY0 -value MAPPED
```

Example 4 The following example illustrates the Tcl command usage for setting the reimplementation effort for a partition.

```
icf_part set_effort -part CGROUP_GRAY0 -value GUIDED
```

Example 5 The following example illustrates the Tcl command usage for setting the bounding box for a partition.

```
icf_part set_bbox -part CGROUP_GRAY0 -width 4 -height 4
```

Example 6 The following example illustrates the Tcl command usage for setting the anchor for a partition.

```
icf_part set_anchor -part CGROUP_GRAY0 -anchor R10C10
```

Example 7 The following example illustrates the Tcl command usage for setting the color for a partition.

```
icf_part set_color -part CGROUP_GRAY0 -color R0G50B255
```

Chapter 22

Glossary

A

.a An .a file is a platform library archive file in LatticeMico™ System that is automatically generated during a platform library build. It is referenced by the application build and derived from the platform library object files.

ABR ABR is an abbreviation for available bit rate, an ATM service type in which the network makes a “best effort” to meet the transmitter’s bandwidth requirements.

AC AC is an abbreviation for alternating current, which is electric current that rises to a maximum in one direction, falls back to zero, rises to a maximum in the opposite direction, and then repeats.

accumulator An accumulator is a register for adding, subtracting, or both.

active Active describes the window or design file that is currently being worked on. An active window has minimize and maximize buttons and a control menu box displayed on the title bar.

Active-HDL Active-HDL is a simulation tool from Aldec that supports Verilog and VHDL simulation and verification.

active high Active high (or High) is a signal whose active state is logic 1.

active low Active low (or Low) is a signal whose active state is logic 0.

activity factor An activity factor is a user-supplied variable in Power Calculator that estimates the percentage of the design that is changing state.

ADC ADC is an abbreviation for an analog-to-digital converter, which is a device that converts continuously varying analog signals to binary code that can be processed by a microprocessor or microcontroller. An ADC can be a single chip or a circuit in a chip.

adder An adder is a combinatorial circuit that computes the sum of two or more numbers.

AES Advanced Encryption Standard is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

air flow Air flow is the movement of air around the device in a package to cool it. It is measured in linear feet per minute (LFM).

algorithm An algorithm is a specified sequence of actions performed to accomplish a task or solve a problem. It especially refers to a recursive computational procedure for solving a problem in a finite number of steps.

ALU ALU is an abbreviation for arithmetic logic unit. See "arithmetic logic unit" on page 2638.

ambient temperature The ambient temperature is the expected operating temperature, in degrees Celsius, of the medium surrounding a device in a package.

analog Analog describes a signal that is continuously variable. The characteristics of an analog signal, such as amplitude or frequency, vary continuously and infinitely. In contrast, the characteristics of a digital signal are divided into a finite number of discrete steps.

ANSI ANSI is an abbreviation for the American National Standards Institute, an industry-sponsored group that sets cooperative standards for companies to follow.

antenna effect The antenna effect is the buildup of charge on a conductor like polysilicon or metal that is not grounded, or electrically connected to silicon, during the polysilicon etch process or the ion implantation process. The voltage may build up on the interconnect to the point where it discharges and inflicts permanent physical damage. For example, damage to the gate oxides.

API API is an abbreviation for application programming interface, which is a language used by a computer application to interact with the operating system.

application build An application build is the output of the managed build process in LatticeMico System, for example, the application executable, application build makefiles, application object files, and required platform library files.

application build makefiles Application build makefiles are makefiles that enable the building of the application in LatticeMico System.

application executable The application executable is a file that results from linking the application and the platform library object file in LatticeMico System. This file is an .elf file that can be downloaded or executed using the GNU GDB debugger.

application object files Application object files are user source object files in LatticeMico System that have been compiled and assembled from the source C files.

architecture Architecture is the common logic structure of a family of programmable integrated circuits. The same architecture can be realized in different manufacturing processes.

arithmetic logic unit An arithmetic logic unit (ALU) is a logic function that performs arithmetic computations, such as addition, multiplication, and comparison operations. The ALU is one component of the central processing unit (CPU).

.asc An .asc file is an ASCII schematic file.

ASC Analog Sense and Control device used with MachXO2 and Platform Manager 2 designs.

ASIC ASIC is an acronym for application-specific integrated circuit, which is a full-custom circuit manufactured for a specific application in which every mask is defined by the customer. It can also be a semi-custom circuit (gate array) where only a few masks are defined. In large volume, ASICs are cheaper than programmable logic devices, but lead times and initial costs are large.

ASSP ASSP is an abbreviation for application-specific standard part, which is an integrated circuit created by using ASIC technology but sold as a standard device on the open market.

.asy An .asy file is an ASCII symbol file.

asynchronous Asynchronous describes an entity that is not synchronized to a shared signal like a clock.

asynchronous logic Asynchronous logic is logic that changes independently of clock changes. It is a signal whose intended function is performed immediately at the point that the signal is asserted without regard to a clock.

asynchronous transfer mode Asynchronous transfer mode (ATM) is a technique in which channel demand determines packet allocation. ATM offers fast packet technology, real time, and demand-led switching for efficient use of network resources. It is also the generic term adopted by ANSI and the ITU-TS to classify cell relay technology within the realm of broadband WANs, specifically B-ISDN.

ATE ATE is an abbreviation for automated test equipment, which refers to testers.

ATM ATM is an abbreviation for asynchronous transfer mode. See “asynchronous transfer mode” on page 2638.

attribute An attribute is an HDL constraint that is attached as text to a design object and interpreted by the Diamond software. An attribute provides information about the object. For example, it might specify where a component in the logical design must be placed in the physical device, or it might specify a frequency constraint for a net that timing-driven place and route will attempt to meet.

attribute window An attribute window is a predefined schematic area on or near a symbol or pin in which attribute values are displayed.

B

back annotation Back annotation is the process of adding post-route timing delays to the post-layout netlist in preparation for post-layout timing simulation. Before implementation, logic path delays are only estimated. The actual wiring delays (that is, the parasitic resistance and capacitance) cannot be known until after placement and routing has been performed. Back annotation substitutes the actual delays for the estimated delays in the netlist.

backplane A backplane is a circuit board containing circuitry and sockets into which additional electronic devices on other circuits boards or cards can be plugged.

bank A bank is an arrangement of identical hardware components that are electrically connected, such as I/O pads.

behavioral design Behavioral design is a technology-independent, text-based design that incorporates high-level functionality and a high-level information flow.

behavioral design method A behavioral design method is a method of defining a circuit in terms of a textual language rather than a schematic of interconnected symbols.

behavioral simulation Behavioral simulation is the same as functional simulation. See “functional simulation” on page 2655.

BER BER is an abbreviation for bit error ratio (or rate), which is the number of erroneous bits divided by the total number of bits transmitted, received, or processed over a stipulated period. BER monitoring is one of the PCS block functions.

.bgn A .bgn file is an FPGA report file showing any DRC errors or warnings associated with generating the bitstream. Running the Bitstream File process in Lattice Diamond’s Process view generates this file. See also “bitgen” on page 2640.

BIDI BIDI is an abbreviation for a bidirectional signal.

bidirectional A bidirectional is a signal or port that can act as either an input or an output.

binary Binary refers to the base 2 numbering system.

binary encoding Binary encoding is a type of state machine encoding that uses the minimum number of registers to encode the machine. Each register is used to its maximum capability.

.bit A .bit file is a binary configuration bitstream data file that is used to program an FPGA device. This file is similar to the raw bit file. See also ".rbt" on page 2683.

bitgen Bitgen is a command-line program that converts a fully routed physical design into configuration bitstream data. It is equivalent to the Bitstream File process in Lattice Diamond's Process view. The program takes a fully routed physical design, in the form of a circuit description (.ncd) file as input and produces a configuration bitstream (bit images).

bitstream A bitstream is a sequence of binary data that contains the configuration information from the physical design defining the internal logic and interconnections of the FPGA, as well as device-specific information from other files associated with the target device. When downloaded to a device, a bitstream configures the logic and programs the device so that the states of that device can be read back.

block A block is a symbol that represents lower-level circuitry; for example, a group of one or more logic functions or a schematic or symbol sheet. A block does not show any details of the circuitry, except for I/O pins. Blocks are most often used to simplify a complex schematic or to allow a large schematic to fit on a single sheet. Blocks usually represent repeated sections of circuitry but can replace any part of a schematic. As a representation of a schematic or symbol sheet, there are four types of blocks: 1) A composite block indicates that the design is hierarchical. 2) A module block is a symbol with no underlying schematic. 3) A pin block represents a schematic pin. 4) An annotate block is a symbol without electrical connectivity that is used only for documentation and graphics.

block check A block check is a physical DRC check in the EPIC software that examines one or more placed or unplaced components and reports any problems with logic, physical pin connections, or programming. It can be performed on selected components or on all components in the design. When you check all components in the design, the block check performs extra tests on the design as a whole (for example, tristate buffers sharing long lines and oscillator circuitry configured correctly) in addition to checking the individual components. You can run the block check separately or with the net check.

block modular design Block modular design is a design strategy that allows multiple designers to work on parts, or "block modules," of a single design independently. It also enables a single designer to make incremental changes with minimal impact to other portions of the design.

block symbol A block symbol is a schematic symbol that represents a schematic or HDL module at the next-lower level of hierarchy.

boundary scan Boundary scan is a method of testing electronic assemblies at the board level by inserting registers and control logic to serially scan in test patterns and scan output data registers. The primary objectives are the testing of chip I/O signals and the interconnections between ICs. It is the method for observing and controlling all new chip I/O signals through a standard interface called a test access port (TAP). The boundary scan architecture includes four dedicated I/O pins for control and is described in IEEE spec 1149.1.

breakpoints Breakpoints are a combination of signal states that are used to indicate when simulation should stop. They enable you to stop a program at certain points to examine the current state and the test environment to determine whether the program functions as expected.

BS See “boundary scan” on page 2640.

BSCAN See “boundary scan” on page 2640.

BSDL BSDL is an abbreviation for Boundary Scan Description Language, which is an IEEE language used to describe the structures for boundary scan testing.

buffer A buffer is an isolation circuit used to insulate sensitive analog or digital circuits from higher power levels or higher current levels in other portions of the chip.

burst mode Burst mode is a type of high-speed data transmission in which multiple transfers of data take place between a master and a slave within the same bus transaction.

bus A bus is a named collection of two or more wires or other buses that carry closely associated signals in an electronic design. Attaching a bus tap to a single wire “promotes” the wire to a bus. In a schematic, a bus looks like a single wire but is displayed in a different color. (The color does not appear until you name the bus.) The individual signals within a bus can be tapped off at any point along the bus, as often as needed. (Schematic Editor does not check to see if you are overloading the bus.)

bus pin A bus pin is a pin used to connect a bus to a symbol.

bus tap A bus tap is the point at which a signal enters or exits a bus.

C

C/C++ SPE C/C++ SPE is an abbreviation for the C/C++ Software Project Environment. See “SPE” on page 2689.

CAM CAM is an abbreviation for content addressable memory. Also known as “associative storage,” it is a memory chip in which each bit position can be compared. In regular dynamic RAM (DRAM) and static RAM (SRAM) chips, the contents are addressed by bit location and then transferred to the arithmetic logic unit (ALU) in the CPU for comparison. In CAM chips, the content is compared in each bit cell, allowing for very fast table lookups.

capacitance Capacitance is the ability of a conductor to store electric charge or a certain number of electrons. It is expressed in farads.

capacitive reactance Capacitive reactance is the opposition to current flow provided by a capacitor. It is measured in ohms and varies inversely with frequency.

carry logic Carry logic is logic that is designed to speed up and reduce the area of counters, adders, incrementors, decrementors, comparators, and subtractors. It is a special interconnect that speeds up the carry path of adders and counters.

CDR CDR is an abbreviation for clock data recovery. See “clock data recovery” on page 2643.

CDT CDT is an abbreviation for C/C++ development tools, which are components, or plug-ins, of the Eclipse development environment on which LatticeMico™ System is based.

cell library A cell library is the collective name given to a set of logic functions as defined by the manufacturer.

cell symbol A cell symbol is a schematic symbol that represents the primitive cells, such as transistors, resistors, and diodes, used to design integrated circuits.

CFI CFI is an abbreviation for Common Flash Interface (CFI) parallel flash memory, which is an open standard jointly developed by a number of chip vendors for a type of EEPROM that stores information without requiring a power source.

chain file A chain file is an .xcf file. See “.xcf” on page 2700.

chip check A chip check is a physical design-rule check that examines a special class of checks for nets, components, or both at the chip level, such as placement rules with respect to one side of the device.

CIB CIB is an abbreviation for a common interface block, which is a switch matrix that can be programmed to connect virtually any routing resource to any input or output of the logic element.

CIP CIP is an abbreviation for configurable interconnect point, which refers to the switching circuitry that connects routing segments on a net. It provides one or more of three basic functions: signal switching, amplification, or isolation. A net running from a PFU or PIC output (source) to a PLC or PIC input (destination) consists of one or more routing segments connected by CIPs. In EPIC, the graphic representations of CIPs are also called nodes.

cleanup routing Cleanup routing is the second stage of routing (also called delay reduction routing). During cleanup routing, the router takes the result of iterative routing, the first stage, and reroutes some connections to minimize the signal delays within the device.

CLKFB divider CLKFB is an abbreviation for feedback clock divider, which is a PLL block that multiplies the output of a PLL. The CLKFB divider is used to divide the feedback signal. Effectively, this multiplies the output clock, because the divided feedback must speed up to match the input frequency into the PLL block. The PLL block increases the output frequency until the divided feedback frequency equals the input frequency. Like the input divider, the feedback loop divider can be set to an integer value of 1 to 16. The input

and output of the feedback divider must be within the input and output frequency ranges specified in the device data sheet.

CLKI divider CLKI is an abbreviation for input clock divider, which a PLL block that divides the output of a PLL. The CLKI divider is used to control the input clock frequency into the PLL block. It can be set to an integer value of 1 to 16. The divider setting directly corresponds to the divisor of the output clock. The input and output of the input divider must be within the input and output frequency ranges specified in the device data sheet.

CLKOK divider CLKOK is an abbreviation for clock divider, which is a PLL block that obtains the divided value of the primary clock. The CLKOK divider feeds the global clock net. It divides the CLKOP signal of the PLL by the value of the divider. It can be set to values of 2, 4, 6, ...126,128.

CLKOP divider CLKOP is an abbreviation for primary clock output divider, which is a PLL block that sets the VCO operating frequency. The CLKOP divider serves dual purposes of squaring the duty cycle of the VCO output and scaling up the VCO frequency into the 420-MHz to 840-MHz range to minimize jitter.

clock A clock is a signal or square waveform used to synchronize the timing of the circuits in a digital integrated circuit. It switches between a high and a low state, usually with a 50-percent duty cycle.

clock boosting Clock boosting is the timing optimization of designs through clock skew scheduling to improve the performance in a synchronous system. Clock boosting is achieved by scheduling clock delay signals at each register to relax critical paths.

clock data recovery Clock data recovery is the process of eliminating channel-to-channel, clock-to-channel skew. It is implemented in the HSI embedded module. The clock signal is encoded in the data stream. This encoding scheme eliminates the need for a separate clock channel and assures that the clock and data are in phase, enabling higher bandwidth at lower cost.

clock enable A clock enable is a binary signal that allows or prevents synchronous logic from changing with a clock signal. When enabled, this control signal permits a device to be clocked and to become active. There are four different states. The two active high states are CE 0 disabled and CE 1 enabled. The two active low states are CE 0 enabled and CE 1 disabled.

clock frequency Clock frequency is the number of complete clock cycles that occur per unit of time. For example, a 50-MHz clock signal cycles 50 million times per second.

clock gating Clock gating is the process of using combinational logic elements, such as an AND gate, to disable the clock signal in unused circuits. Clock gating prevents power dissipation due to unnecessary charging and discharging of the unused circuits.

clock input path A clock input path is a path that starts either at an input of the chip or at the output of a flip-flop, latch, or RAM and ends at any clock pin

on a flip-flip or latch enable. The clock input path time is the maximum time required for the signal to arrive at the flip-flop clock input. Clock input paths help to determine system-level design timing.

clock latency Clock latency is the number of clock cycles that it takes data to propagate from input to output.

clock pulse A click pulse is a signal that synchronizes the operation of an electronic system. Clock pulses are continuous, precisely spaced changes in voltage.

clock skew Clock skew is the time differential between two or more destination pins in a path.

clock tree A clock tree is a network of clock signals distributed throughout a design. An optimal clock tree replaces high-fanout nets with a balanced tree of buffers to minimize clock skew.

clock-tree synthesis Clock-tree synthesis is the process of generating an optimal clock-tree network by inserting a tree of buffers to replace a high-fanout net, balancing the clock arrival times at the leaf nodes to minimize clock skew.

code-relocator code Code-relocator code is code in LatticeMico™ System that copies the software application code to a destination memory and jumps to the application start address to run the application.

combinational function A combinational function is a logic circuit whose output values depends on the current combination of values on its inputs at any given moment. See also “sequential function” on page 2687.

comparator A comparator is a logic function used to compare sets of binary values to see if they contain the same data.

compiling Compiling is the process of changing the design entry format into Boolean equations, which serve as input to simulation and device implementation programs. In general, compiling a design involves running every process after design entry. These processes include compiling and optimizing steps that can be performed on a single source or on the entire design. The Lattice Diamond software accept several design entry formats. With the exception of EDIF, all designs must be either synthesized or compiled before being placed and routed.

component A component is an instantiation or symbol reference from a library of logic elements that can be placed on a schematic. Components are constructed using primitive logic gates or primitives. For EPIC users, a component more specifically refers to a logical configuration that will, at some point, go into a site. Examples of components are logic blocks, I/O blocks, tristate buffers, GSRs, PCMs, and oscillators.

component description file See “.xml” on page 2700.

component information structure declaration The component information structure declaration is specified as part of the component

description (.xml) file in LatticeMico System and is copied into the platform description (.msb) file by the Mico System Builder (MSB). Each component in the platform is represented in the .msb file. The component's information in the .msb file includes the details about the component's source files that will need to be included in the build process. The information is then extracted from the .msb file by the build process and put into the DDStructs.h file. Each unique component must have its own unique component information structure defined within its component description file.

component instance declaration The component instance declaration declares presence of an instantiated structure for those component instances that have a corresponding information structure in LatticeMico System. It originates in the component description (.xml) file.

configuration Configuration is the process of loading design-specific bitstreams into one or more programmable devices to define the functional operation of the logical blocks, their interconnections, and the chip I/O.

congestion Congestion is the ratio of the number of wires that are actually routed across the edge of a cell to the capacity of the edge to accept them

constrained path A constrained path is a path that has a timing preference or requirement ascribed to it.

constraint file A constraint file is a file that specifies design constraint (location and path delay) information in textual form. In Lattice Diamond, a constraint file can be a logical preference file or a synthesis constraint file. See also ".lpf" on page 2667and "synthesis constraint file" on page 2692.

constraints Constraints are directives that are applied by the placement and routing programs to meet user objectives. Constraints can be declared as FPGA attributes in the HDL, schematic, or EDIF source files; or they can be declared as preferences in the logical preference file (.lpf). For designs that use the Lattice Synthesis Engine (LSE), constraints can also include Synopsys® Design constraints (SDC) that are directly interpreted by the synthesis engine.

constructive placement Constructive placement is the first stage of placement. In this stage, PAR places components into sites on the basis of such factors as the constraints specified in the input file, the length of connections, the available routing resources, the cost tables that assign random weighted values to each of the relevant factors. See also "optimizing placement" on page 2674.

core A core is a relatively large block that implements a general-purpose logic function and is used as a building block in chip design. Examples of cores are microprocessor and DSP cores. Cores are also called IP (intellectual property).

counter A counter is 1) a logic function composed of flip-flops that stores count values. The number of value states through which a counter sequences before returning to its original value is called the modulus. For example, a modulus that counts from 0 to 15 (decimal equivalent to binary) is called a modulo-16, or mod-16, counter. 2) A mechanism connected to the pattern

comparator in the Reveal core that specifies the number of times that the sampled trigger bus data must match the selected pattern in order for an event to occur.

coupling capacitance Coupling capacitance is the parasitic capacitance between two signals. It is used for allowing only certain AC signals to pass to other circuits.

CPLD CPLD is an abbreviation for complex programmable logic device, which is an erasable programmable logic device that can be programmed with a schematic or a behavioral design. CPLDs constitute a type of complex PLD based on EPROM or EEPROM technology. They are characterized by an architecture offering high speed, predictable timing, and simple software. The basic CPLD cell is called a macrocell, which is the CPLD implementation of a GLB. It is composed of AND gate arrays and is surrounded by the interconnect area.

critical path The critical path is the path through a circuit that determines the maximum speed at which the circuit can operate. The critical path is a signal in a section of combinatorial logic that limits the speed of the logic. Storage elements begin and end a critical path, which may include I/O pads.

cross-probing Cross-probing is 1) the inter-process communication between the synthesis or simulation results and the source code from which it originated 2) selecting an element in one view in Lattice Diamond and displaying the corresponding logical or physical element in a different view.

crosstalk Crosstalk is a charge injected on a net by capacitive or inductive coupling when coupled signals switch voltage.

CSR CSR is an abbreviation for a control and status register, which is a register in most CPUs that stores additional information about the results of machine instructions, for example, comparisons. It usually consists of several independent flags, such as carry, overflow, and zero. The CSR is mainly used to determine the outcome of conditional branch instructions or other forms of conditional execution.

.csv A .csv file is a file in the comma-separated values delimited data format, in which fields are separated by the comma character and records are separated by newlines. Fields that contain a comma, a newline, or double quotation marks or that start or end with white spaces to be preserved are enclosed in double quotation marks.

cycle stealing Cycle stealing is the timing optimization of designs through clock skew scheduling to improve the performance in a synchronous system. It is achieved by scheduling clock delay signals at each register to relax critical paths.

D

DAC DAC is an abbreviation for a digital-to-analog converter, which is a device that converts digital (usually binary) code to analog signals. A DAC can be a single chip or a circuit in a chip.

daisy chain A daisy chain refers to multiple FPGAs connected in a series that can be configured at the same time. A daisy chain uses a lead FPGA and one or more FPGAs configured in slave serial mode. The lead FPGA can be configured in any mode except slave parallel mode.

database See “internal database” on page 2660.

data capture mode Data capture mode is a setting in Reveal Inserter that specifies how the trace buffer in Reveal Logic Analyzer captures the trace data. It can capture data around a single trigger event or multiple trigger events.

dataset A dataset is a collection of the settings, such as trace signals, trigger signals, sample clock, and trigger output signal, for each of the logic analysis cores in Reveal Inserter and Reveal Logic Analyzer. Each dataset can include up to 16 logic analysis cores, one of which is reserved for internal use.

DC DC is an abbreviation for direct current, which is electric current that flows in one direction.

DDR DDR is an abbreviation for a double data rate, a memory interface in an SDRAM that captures data on both the rising and falling edges of the clock, doubling the performance of a single data rate interface.

debug insertion Debug insertion is the process of placing logic analysis cores into a design to debug it. Debug insertion gives you access to internal nodes inside the device so that you can observe their behavior. The Reveal Inserter application in Lattice Diamond is used to configure these cores and Reveal Logic Analyzer to debug the design.

.ddt A .ddt file is a file created by the Deployment Tool when saving a deployment.

debugging Debugging is the process of reading back or probing the states of a configured device to ensure that the device is behaving as expected while in circuit. Debugging in software is the process of locating and reducing the errors in the source code (the program logic). Debugging in hardware is the process of finding and reducing errors in the circuit design (logical circuits) or in the physical interconnections of the circuits.

decimal Decimal refers to the base 10 numbering system.

decoder A decoder is a symbol that translates n input lines of binary information into 2n output lines. It is the opposite of an encoder.

decoupled capacitance Decoupled capacitance is capacitance used to eliminate high-frequency noise from the DC voltage supply to prevent unwanted signals from passing between circuits.

delay Delay is the time associated with a logic function or timing preference placed on a component, net, or path. For example, delay calculations can measure the time that it takes a signal from a driver pin to reach a load pin. Typically, delays are given in nanoseconds. See also “pin-to-pin delay” on page 2677 and “point-to-point delay” on page 2678 delay.

delay reduction routing Delay reduction routing is the same as cleanup routing. See “cleanup routing” on page 2642.

demo board, demonstration board A demo, or demonstration, board is the same as an evaluation board. See “evaluation board” on page 2652.

density Density is the number of gates on a device.

Deployment Tool Deployment Tool is a stand-alone tool available from the Diamond Accessories. The Deployment Tool graphical user interface (GUI) is separate from the Diamond design environment. A four-step wizard allows you to select deployment type, input file type, and output file type.

derating Derating is the practice of reducing the rating of a device, such as its current rating, to provide an additional margin of safety or reliability when the device operates under unusual or extreme conditions.

design implementation Design implementation is the process of running a series of specific programs that result in the creation of a completed design based on user design specifications. The implementation process begins with checking design syntax. Language is then translated and optimized. The design logic is then mapped, placed, and routed.

design translation Design translation is the process of converting an input netlist into primitive terms that the software can understand and manipulate. The Translate Design process in Lattice Diamond converts the input EDIF file to a Native Generic Database format and builds the database, producing an .ngd file.

design methodologies Design methodologies are techniques used to enter a design, either through behavioral design or schematic entry.

design specification Design specification is the top level of a design used to define its function. The specification function is created in terms of behavioral or structural primitives. The two methods of entering a design are graphical descriptions (schematics) and textual descriptions (HDL).

detailed routing Detailed routing is the process of routing regions by assign each net specific tracks within the region.

deterministic jitter Deterministic jitter is jitter reproducible within a given system under controlled conditions. It is also known as bounded jitter. See also “jitter” on page 2662.

device A device is 1) an integrated circuit or other solid-state circuit formed in semiconducting materials during manufacturing 2) A particular architecture, such as a LatticeECP device.

device driver files Device driver files are the source .c and .h C/C++ files in LatticeMico System that contain driver code that is compiled into object files during the software build.

device family A device family is a group of Lattice Semiconductor devices based on the same architecture. Examples of device families are LatticeEC, LatticeXP, and MachXO.

device model A device model is a VHDL description of the internal and external views of a digital device, including the structure and the communication interface of the device with its environment.

Device View Device View is an integrated tool in Diamond that provides a categorized list of device resources based on the target device. It enables you to cross-probe selected components to Floorplan View and prohibit sites that you want excluded from placement and routing. See also “Floorplan View” on page 2654.

device utilization Device utilization is the relative density at which the device is packed or utilized. More specifically, device utilization refers to the percentage of available PFU sites that have been utilized on a device. The term device density, on the other hand, typically refers to the percentage of available logic gates that are being utilized on a device.

dielectric Dielectric is the insulating material between two metal plates, especially the two metal plates of a capacitor.

digest (message digest) a message digest is a cryptographic hash function containing a string of digits created by a one-way hashing formula.

digital signal processing Digital signal processing is the analyzing or changing of digital signals that have been converted from analog signals, such as sound or video.

DIP DIP is an abbreviation for dual in-line package, a through-hole-style plastic chip package.

DLL DLL is an abbreviation for a delay-locked loop, which is a digital circuit used to perform clock management functions on and off the chip.

.dly The .dly file is the delay report file, which is an FPGA place-and-route output file containing delay information for each net in the design. This file is generated when you use the -y option of the par program, which performs the placement and routing.

downloading Downloading is the process of configuring or programming a device by sending bitstream data to the device through a download cable. Configuration data from bitstream or PROM files is transmitted to the download cable through the host computer’s serial communications port.

DPD DPD is an abbreviation for dedicated power down, a mode on zero-power devices. DPD is also refers to the dedicated power-down enable pin.

DPP DPP is an abbreviation for dedicated power pin, the control pin on dedicated zero-power devices. The term DPP also refers to dedicated power-down devices.

DQS DQS is a bidirectional data strobe signal used by the DDR interface in an SDRAM for high-speed operation.

DQSDLL DQSDLL is a signal that generates a 90-degree phase shift required for the DQS signal in the DDR interface of an SDRAM.

DRC DRC is an abbreviation for design rule check, which is a series of tests used to discover logical and physical errors in the design.

driver pin A driver pin is a component pin capable of transmitting a signal to one or more of its designated load pins. Output pins and tristate pins are considered drivers.

DSP DSP is an abbreviation for digital signal processing. See “digital signal processing” on page 2649.

duty cycle The duty cycle is the ratio of the time that a component, device, or system is operated to the time that it is at rest. The duty cycle can be expressed as a ratio or as a percentage.

E

E2 E2 is another term for electrically erasable, which refers to the technology of non-volatile memory using isolated capacitors that can store charge.

E^2CMOS E^2CMOS is an abbreviation for electronically erasable complementary metal oxide semiconductor, which is a proprietary Lattice Semiconductor process considered the preferred process technology for PLDs because of its inherent performance, reprogrammability, and testability benefits.

EBR EBR is an abbreviation for embedded block RAM.

Eclipse Eclipse is an open-source platform that provides application frameworks for software application development. The LatticeMico System interface is based on the Eclipse environment. See also “LatticeMico System” on page 2664.

ECO ECO is an abbreviation for engineering change order, which is the process of rebuilding a design after limited engineering changes have been made to it.

ECDSA Elliptical Curve Digital Signature Algorithm (ECDSA) is the elliptical curve analogue of the DSA.

.edf An .edf file is a file containing an EDIF netlist. It is the default netlist file that is generated by the Precision synthesis tool.

edge clock An edge clock is a clock that resides at the perimeter of an FPGA. Edge clocks are used for very localized clocking of I/O elements. Because edge clocks incur significant skew (like secondary clocks), they are not suited for clocking large buses. Edge clock routing resources reside on separate mask layers of the FPGA in order not to interfere with other signals or clocking resources in the design.

edge-sensitive signal An edge-sensitive signal is an input that only affects a function when it transitions from one logic value to another. See also "level-sensitive signal" on page 2665.

.edi An .edi file is an internal netlist file generated by Diamond. The file extension must be changed to .edf or .edn in order to import it into a project.

EDIF EDIF is an acronym for Electronic Design Interchange Format, which is a format used to exchange design data between different ECAD systems. It is designed to be written and read by computer programs that are constituent parts of EDA systems or tools. Its syntax, which is similar to LISP, has been designed for easy machine parsing. It is an essential part of the Electronic Industries Alliance's (EIA) service to the electronics industry. Since its inception over thirteen years ago, EDIF has become a vital part of the Electronic Design Automation (EDA) industry. The Diamond software supports EDIF Version 2 0 0.

edif2ngd Edif2ngd is a command-line program that, together with ngdbuild, converts netlists into a generic database that uses native primitive terms that the Diamond software can implement. It is equivalent to the Translate Design process.

.edn An .edn file is a file containing an EDIF netlist. It is the default netlist file that is generated by the Synplify synthesis tool.

EEPROM EEPROM is an acronym for electrically erasable programmable read-only memory, which is a programmable memory chip that holds its content without power and can be re-programmed in place without exposure to ultraviolet light.

EFB EFB is an abbreviation for embedded function block, an IP module with hardened functions such as SPI, I2C, and Timer/Counter. The EFB block is connected to the device from a Wishbone bus. See also "Wishbone Bus" on page 2699.

Effective Theta JA Effective Theta JA is a measurement of the sum of all the package and board thermal resistances outside of a JEDEC-defined environment. It indicates how well the heat dissipates from the die to the ambient (air) for a particular thermal network.

effort level Effort level refers to how hard the implementation process tries to place a design. The effort level settings are: 1) High, which provides the highest quality placement but requires the longest execution time. Use high effort on designs that do not route or do not meet your performance requirements. 2) Normal, which is the default effort level. It provides the best trade-off between execution time and high-quality placement for most designs. 3) Low, which provides a lower-quality placement but requires the shortest execution time.

electromigration Electromigration is the erosion of a metal conductor by the flow of electrons in an integrated circuit because of Joule heating and high current density.

Electronic Design Interchange Format See "EDIF" on page 2651.

.elf An .elf file is a file in executable linked format that contains the software application code written in C/C++ SPE.

embedded block RAM An embedded block RAM is a RAM cell in an FPGA that can be configured for RAM, ROM, FIFO, CAMs, and so forth.

EPIC EPIC is an acronym for Editor for Programmable ICs, which is a Lattice Diamond stand-alone application for displaying and configuring FPGAs. It can be used for placing and routing critical components before running the automatic place-and-route tools on an entire design. It can also be used for manually finishing the placement and routing if the routing program was unable to route the design to completion. EPIC enables both automatic and manual component placement and routing. It also interacts with the preference (.prf) file to read, write, and undo certain preferences.

.epl An .epl file is an EPIC Device Editor log file that tracks all commands performed on the design file in an active file session. The commands are written into a temporary command log file called <*design_name*>.epl in the same directory where the design is stored.

EPROM EPROM is an acronym for electrically programmable read-only memory, which is a programmable memory chip that holds its content without power and must be placed under ultraviolet light for erasure before reprogramming.

.err The .err file is an error output file.

evaluation board An evaluation board is a platform on which to evaluate, test, and debug user designs.

event An event in Reveal Inserter and Reveal Logic Analyzer is a trigger bus pattern that occurs a programmable number of times.

event counter An event counter in Reveal Inserter and Reveal Logic Analyzer is the maximum size of the count of all the trigger expression outputs combined.

.exo The .exo file is a PROM-formatted configuration file in Motorola EXORmacs format.

EXORmacs EXORmacs is a PROM file format from Motorola.

external clock An external clock is the system clock that is used on the target board during synchronous mode debugging. To use an external clock, connect the system clock to the CLKI pin and connect the download cable CLKO pin to the system clock loads.

external net An external net is a net outside of a macro instance connected to one of the macro's external pins. Part of an external net may lie within the macro, if the macro library file contained a net that was connected to an external pin.

External Netlist Format External Netlist Format is a netlist format created by Xilinx. See also ".xnf" on page 2700.

external pin An external pin is a macro pin used to connect the components in an instantiated macro to other components in a design (outside of the macro).

F

fab Fab is an abbreviation of fabrication, which is the process of wafer fabrication or the fabrication facility itself.

fall time Fall time is the time it takes the falling edge of a pulse to go from 90 percent of peak voltage to 10 percent of peak voltage.

falling edge A falling edge is a signal transition from a logic 1 to a logic 0. It is also called a negative edge.

fanin Fanin is the maximum number of digital inputs that a single logic gate can accept.

fanout Fanout is the maximum number of digital logic gates that can be driven by the output of a single logic gate.

.fdc An .fdc file is a constraint file used with the Synplify Pro synthesis tool. The file uses the same constraint format as the Synopsys Design Constraint (SDC) language and replaces the .sdc files used in earlier versions of Synplify Pro. FDC is an abbreviation for FPGA Design Constraint. See also "SDC" on page 2686 and "synthesis constraint file" on page 2692.

FDM FDM is an abbreviation for frequency division multiplexing, which is a technology that transmits multiple signals simultaneously over a single transmission path, such as a cable or wireless system. Each signal path travels within its own unique frequency range (carrier), which is modulated by the data (text, voice, video, and so forth).

.fdo The .fdo file is a batch file output by ModelSim/Questa.

FEC FEC is an abbreviation for forward error correction, a system of error control for data transmission in which the receiving device has the capability of detecting and correcting any character or code block that contains fewer than a predetermined number of symbols in error. FEC is performed by using a predetermined algorithm to add bits to each transmitted character or code block.

feedthrough A feedthrough is a signal net within a cell that can be connected to the top-level routing at multiple locations simultaneously. It is commonly used in cells intended to be assembled by abutment to form a continuous net passing through the cells. The feedthrough may or may not connect to the circuitry in the cell.

FFT FFT is an abbreviation for fast fourier transform, which is an algorithm for converting data from the time domain to the frequency domain. It is often used in signal processing.

FIFO FIFO is an acronym for first in, first out, a method of data storage in which data is retrieved in the same order as in was put in.

FIR filter A FIR (finite impulse response) filter is a filter whose output is a weighed sum of the current and past inputs. It is one of two types of filters used in digital signal processing. The IIR (infinite impulse response) filter is the other. FIR filters are used more frequently than IIR filters because they are stable; require no feedback; can be designed to be linear phase filters, which delay the input signal but do not distort its phase; and are simple to implement. However, FIR filters are less efficient than IIR filters.

flash memory Flash memory is a type of programmable chip that retains data even when the power is turned off.

flat design A flat design is a design composed of multiple sheets at the top-level schematic.

flattening Flattening is the process of resolving all the hierarchy references in a design. If a design contains several instantiations of a logic module, the flattened version of that design duplicates the logic for each instantiation. A flattened design still contains hierarchical names for instances and nets.

flip chip A flip chip is a surface mount chip technology in which the chip is packaged in place on the board and then underfilled with an epoxy. A common technique for attachment is to place solder balls on the chip, “flip” the chip over onto the board, and melt the solder.

flip-flop A flip-flop is a simple two-state logic buffer activated by a clock and fed by a single input working in combination with the clock. The states are high and low. When the clock goes high, the flip-flop works as a buffer as it outputs the value of the D input at the time the clock rises. The value is kept until the next clock cycle (rising clock edge). The output is not affected when the clock goes low (falling clock edge).

Floorplan View Floorplan View is a view in the Diamond that provides a convenient large-component layout of your design. It shows placement and routing and provides toolbar and menu commands for displaying UGROUPS

and REGIONs, utilized and reserved resources, and connections. Floorplan View allows you to create REGIONs and bounding boxes for UGROUPs, and reserve resources on the layout.

floorplanning Floorplanning is the process of choosing the best grouping and connectivity of logic in a design. It is also the process of manually placing blocks of logic in an FPGA device, where the goal is to increase density, routability, or performance.

flow Flow is an ordered sequence of processes that are executed to produce an implementation of a design.

f_{MAX} f_{MAX} is 1) the maximum number of times per second that a chip can generate logic functions. 2) the symbol for the maximum operating clock frequency, expressed in hertz. The maximum frequency is a function of the longest combinatorial delay between registered elements. See also “clock frequency” on page 2643.

FPGA FPGA is an abbreviation for field-programmable gate array, a type of gate array that is programmed by the user rather than by a semiconductor manufacturer. Most FPGAs are based on SRAM technology, although there are some flash and antifuse versions. FPGAs are similar in function to CPLDs but have different architectural structures.

.fp1 A .fp1 file is an ORCAstra configuration file. This file stores an FPGA configuration as defined in a main Visual Window. See also “ORCAstra” on page 2674 and “Visual Window” on page 2699.

.fpm A .fpm file is an ORCAstra macro file. Macros can be used to automate some operations in ORCAstra. See also “ORCAstra” on page 2674.

frame See “trace frame” on page 2695.

fringing capacitance Fringing capacitance is the same as peripheral capacitance. See “peripheral capacitance” on page 2676.

functional simulation Functional simulation is the process of identifying logic errors in a design before the design is implemented in a device. Because timing information for the design is not available, the simulator tests the logic in the design using unit delays. Functional simulation is usually performed on designs that are entered using a hardware definition language (HDL). It is usually performed at the pre-synthesis stage of the design process. Functional simulation verifies that the HDL code describes the desired design behavior.

fuse A fuse, or fusible link, is a metal wire or strip that melts when a current reaches a specified temperature, opening the circuit in which it is embedded to protect the circuit from an overcurrent condition.

fuse map A fuse nap is a design file in which the fuse data is already prearranged in exactly the same format as the physical layout of the fuse array of the device.

G

gate A gate is 1) an integrated circuit composed of several transistors and capable of representing any primitive logic state, such as AND, OR, XOR, or NOT inversion conditions. Gates are also called digital, switching, or logic circuits. 2) the terminal in a field effect transistor (FET) that applies voltage to cause the transistor to switch.

gate array A gate array is a chip that contains logic elements that have not been interconnected. When the metal layers that provide the connections are adhered to the chip, the masking stage is completed and the chip is finished. This final stage is less costly than designing a chip from scratch. The gate array is composed of basic cells containing a number of transistors and resistors, depending on the vendor. You use a cell library (gates, registers, and so forth.) and a macro library (more complex functions) to design the chip, and the vendor's software generates the masks that connect the transistors.

gated clock See "clock gating" on page 2643.

GDB GDB is an abbreviation for GNU GDB Debugger, which is a source-level debugger based on the GNU compiler. It is part of the C/C++ SPE debugger. It is connected to the various launch configurations connected to ISS or ispVM.

generic database file A generic database file is an .ngd file. See ".ngd" on page 2673.

global routing Global routing is the process of establishing a routing plan for each net. Sections of each net are assigned to specific routing regions according to a particular objective function, for example, total wire length.

glue logic Glue logic refers to simple logic gates used to wire more complex functions together.

GNU Compiler Collection (GCC) The GNU Compiler Collection (GCC) is a set of programming language compilers. It is free software produced by the GNU Project.

graphic symbol A graphic symbol is a schematic symbol that adds information that is not part of the circuitry, such as tables and notes.

ground bounce Ground bounce is an increase in voltage that occurs on the ground network of an integrated circuit because of the current flowing through the resistance of the ground network itself. It is caused by a large number of outputs switching simultaneously.

GROUP A GROUP is a set of FPGA design elements specified with the DEFINE GROUP preference. The GROUP definition enables you to place timing constraints on grouped elements using preferences such as MAXDELAY and MULTICYCLE.

GSR GSR is an abbreviation for a global set/reset signal. In LatticeECP/EC, LatticeXP, and LatticeSC devices, a low signal on the global set reset (GSR) input pin of the GSR component initializes all flip-flops and latches.

guided mapping Guided mapping is an incremental flow that enables you to use a previously generated mapped or routed .ncd file to guide the remapping of the design in a subsequent run after minor changes have been made. Guided mapping preserves the original mapped design while processing just the changes. See also ".ncd" on page 2672.

guided place & route Guided place & route is an incremental flow that enables you to use a previously placed and routed .ncd file to guide subsequent placement and routing of the design after minor changes have been made. Guided place & route preserves the original placed and routed design while processing just the changes. See also ".ncd" on page 2672.

H

HAL HAL is an acronym for hardware abstraction layer, which is the programmer's model of the hardware platform. It enables you to change the platform with minimal impact to your C code.

hard macro A hard macro is a macro that has already gone through internal placement and routing and is simply pasted down on the die. Traditional design reuse is based on hard macros. Such macros change their performance very little when they are moved from one design to another, so the industry has assumed that they require little timing analysis and validation. But hard macros are completely inflexible--their internal logic cannot be altered, even to the extent of changing the size of a buffer or the sense of a logic signal. For this reason and others, most IP providers have focused on soft macros. See also "soft macro" on page 2688.

hardware debugger module The hardware debugger module is a component of C/C++ SPE that is used to find problems in the software application.

hardware platform A hardware platform is the embedded microprocessor in an SoC (system on a chip) design and the attached components, buses, component properties, and their connectivity.

HDL HDL is an abbreviation for hardware description language, which is a textual coding language that describes circuits in a technology-independent manner using a high level of abstraction. The two most widely accepted HDLs are VHDL and Verilog.

heat sink A heat sink is any material or object that dissipates unwanted heat from a device by absorbing it and conducting it away to a surface from which it dissipates into its surroundings.

hexadecimal Hexadecimal refers to the base 16 numbering system.

HGROUP HGROUP is a hierarchical group, an HDL attribute that is used for grouping components that are to be instantiated multiple times. During synthesis, the HGROUP is expanded into individual instances. Each instance is then displayed as a unique UGROUP in the Group sheet of Spreadsheet View. The design mapper translates the HGROUP instances into PGROUPS and includes a hierarchy identifier for each, which enforces strict hierarchical control.

hierarchical design Hierarchical design is a design with more than one level. A large, complex design is typically broken into components, or modules. Circuitry for a specific function or interface can be abstracted as a separate module. The design represented by the module is said to be one level below the design in which the module appears. Or, the design is one level above the module.

hold time The hold time is the period after the clocking edge during which the data input to a latch or flip-flop must be held constant in order to guarantee that the latched data is correct.

HSI HSI is an abbreviation for high-speed interface circuits, which are included in some Lattice Semiconductor devices.

HTML HTML is an abbreviation for Hypertext Markup Language, which is a markup language used to create documents for the Worldwide Web.

hysteresis Hysteresis is the delay between the cause and the observed effect. In Schmitt trigger circuits, it is the difference between the upper and lower trigger points.

I

I2C I2C is an abbreviation for inter-integrated circuit, a multi-master serial single-ended computer bus that attaches low-speed peripherals to a motherboard, embedded system, cellphone, or other electronic device.

I/O blocks I/O blocks are the input/output logic of a device containing a pin.

I/O marker An I/O marker is an indicator that identifies a net name as a device input, output, or bidirectional signal. It establishes net polarity (direction of signal flow) and indicates that the net is externally accessible.

I/O pads I/O pads are the input/output pads that interface between the design logic and the pins of the device.

IBIS model An IBIS, or I/O Buffer Information Specification (IBIS), model is a standard simulation format for modeling the behavior of a digital integrated circuit's input/output (I/O) pins and buffers.

IBUF IBUF is an abbreviation for input buffer, which is an input I/O buffer from a pin to the global routing pool.

IEEE IEEE is an abbreviation for Institute of Electrical and Electronic Engineers, an engineering society.

IEEE 1164 Standard The IEEE 1164 Standard is the standard for Verilog HDL supported by Open Verilog International.

impedance Impedance is the sum of all resistance and reactance of a circuit to the flow of alternating current. It is the opposition that a component in a circuit offers to the flow of alternating current at a given frequency. If the frequency changes, the impedance changes, too. It is similar to resistance but applies to AC circuits.

implementation Implementation is 1) The synthesis and translation, mapping, placement, and routing phases of the design process. See “design implementation” on page 2648. 2) A version of a design project in Lattice Diamond. Each implementation defines the design structural elements and is associated with a specific strategy. See also “strategy” on page 2691.

implementation engine The implementation engine is a PAR multi-run job option that enables you to use multiple machines (nodes) that are networked together, significantly reducing the total amount of time to completion.

indelay Indelay is extra routing delay inserted in MachXO devices on the path from the input pad if you set a hold-time constraint.

inductance Inductance is 1) A property of an electric circuit by which a changing magnetic field creates an electromotive force, or voltage, in that circuit or a nearby circuit. 2) A property of an electric circuit that opposes any change in current.

inductive reactance Inductive reactance is the opposition to the flow of AC current produced by an inductor. It is measured in ohms and varies in direct proportion to the frequency.

.ini (epic) An .ini file (EPIC) is an EPIC Device Editor initialization file. During installation, an EPIC initialization file named epic.ini is placed in the software that is specific to each supported architecture in $FOUNDRY/data, where $FOUNDRY is the path pointed to/defined by the $FOUNDRY environment variable.

.ini (epicuser) The .ini file (EPICUSER) is a file that enables you to customize an epic.ini file. You create the epicuser.ini file in your home directory. This file can be used to modify or add to the epic.ini file so that it performs any commands that you want when the EPIC window opens.

input pad registers and latches Input pad registers and latches are D-type registers or latches located in the I/O pad sections of the device.

inrush current Inrush current is the peak instantaneous current drawn by a power supply when it is first turned on.

insert debug See “debug insertion” on page 2647.

instance An instance is an occurrence or implementation of a specific gate or hierarchical element in a design or netlist (for example, a component, net, site, or macro). The term “symbol” often describes instances in a schematic drawing. Instances are interconnected by pins and nets. Pins are ports through which connections are made from an instance to a net. A design that is flattened to its lowest-level constituents is described with primitive instances.

instantiation Instantiation is the process of placing a symbol that represents a primitive or a macro in a design or netlist. It is also used as a synonym for “instance.”

interconnect Interconnect refers to the physical wires and vias that electrically connect gates and other components in an integrated circuit.

internal database An internal database is the database generated in Diamond after you have created a project and imported the design files. The input for this database is an EDIF file, which can be generated from a standalone synthesis tool and imported into Diamond or created automatically from a Verilog or VHDL design using the integrated synthesis flow within the Diamond software. When you run the Translate Design process, Diamond converts the EDIF file to a Lattice Semiconductor internal database. If the design uses parameterized modules and IP cores (Lattice Semiconductor LPMs and LatticeCOREs), or user firm macros, the cores are expanded in this process.

internal net An internal net is a net within the macro library file that does not have a connection to any of the macro’s external pins.

IOB IOB is an abbreviation for an input/output block.

.ior An .ior file is an I/O Timing Analysis Report that provides worst case I/O summaries for a placed and routed design. For each input data port in the design, it provides the setup and hold time requirements; for each output port, it provides the min/max clock-to-out delay. The .ior file is generated when you run the I/O Timing Analysis process in the Lattice Diamond Process view.

iotiming Iotiming is a command-line program that generates the I/O Timing report (.ior) for a placed and routed design. It is equivalent to the I/O Timing Analysis process in the Lattice Diamond Process view.

IP core An IP, or intellectual property, core is a block of logic that can be used in making ASICs and FPGAs. Examples of IP cores are UARTs, CPUs, ethernet controllers, and PCI interfaces.

IPexpress IPexpress is an application that enables you to generate parameterized modules and IP cores from templates supplied by Lattice Semiconductor and third-party vendors.

.ipx An .ipx file is a manifest file. This file is loaded into IPexpress so that modifications can be made to the module.

IR drop IR drop is a reduction in voltage that occurs on the VDD network of an integrated circuit because of the current flowing through the resistance of the VDD network itself.

IRQ IRQ is an abbreviation for interrupt request, which is the means by which a hardware component requests computing time from the CPU. There are 16 IRQ assignments (0-15), each representing a different physical (or virtual) piece of hardware. For example, IRQ0 is reserved for the system timer, while IRQ1 is reserved for the keyboard. The lower the number, the more critical the function.

ISC ISC is an abbreviation for In System Configuration, a data file standard based on the IEEE 1532 standard for programming one or more compliant devices concurrently while they are mounted on a board or embedded in a system.

ISO ISO is an abbreviation for the International Standards Organization, which is an international collaboration of many different organizations that set industry standards for manufacturing companies. In the U.S., the American National Standards Institute is an industry-sponsored group that belongs to ISO. See also “ANSI” on page 2637.

ISO9000 ISO9000 is a standard of quality for manufacturing companies. See also “ISO” on page 2661.

ISP ISP is an abbreviation for in-system programmable, a four- or five-wire serial programming scheme with internally generated programming voltages. There are two types in Lattice Semiconductor devices: Lattice ISP, and IEEE 1149.1 TAP/BSCAN programming.

ispClock ispClock is a family of Lattice Semiconductor analog devices that can be programmed in-system to generate multiple clock frequencies, compensate each output for differences in clock trace lengths, match trace impedances and drive clock nets with different signaling requirements. Programmable features of ispClock devices can be configured using the Lattice PAC-Designer® software. See also “PAC-Designer” on page 2674.

ispPAC ispPAC (Programmable Analog Circuits) is a group of Lattice Semiconductor programmable analog devices that offers WYSIWYG analog design results. Programmable features can be configured using the Lattice PAC-Designer software. ispPAC devices include Power Manager, Platform Manager, and ispClock. See also “PAC-Designer” on page 2674.

ispSTREAM ispSTREAM is a utility that converts JEDEC files to a bit-packed format, which consumes one-eighth the storage space of the original file. See also “JEDEC” on page 2662.

ispVM System ispVM System is Lattice Semiconductor’s second-generation device download technology, which is used to program, or configure, JTAG programmable devices. It is based on the concept of a portable virtual machine which provides multi-vendor programming support, improved performance, and increased functionality.

ISR ISR is an abbreviation for interrupt service routine, which is a set of instructions in an operating system that executes in response to an interrupt.

ISS ISS is an abbreviation for instruction set simulator, which is a simulation model that imitates the behavior of a microprocessor. Often included in a debugger such as GDB, an ISS enables you to develop and debug a software program while the target hardware is still under development

iterated instance An iterated instance is a single symbol instance that represents multiple instances in a parallel connection.

iterative routing Iterative routing is the first stage of routing, where the router performs an iterative procedure to converge on a solution that routes the design to completion or minimizes the number of unrouted nets.

J

JEDEC JEDEC is an acronym for Joint Electron Design Engineering Committee, which is a group that sets standards for programmable devices. PLD companies refer to programming files as JEDEC files.

jitter Jitter refers to small rapid variations in a waveform resulting from fluctuations in the voltage supply. It can introduce errors and loss of synchronization. More jitter is encountered with longer cables, cables with higher attenuation, and signals at higher data rates. Jitter is also called phase jitter, timing distortion, or intersymbol interference.

Joule heating Joule heating is the excessive heating of a segment of wire because of high currents. It is a cause of electromigration.

.job A .job file is a Model 300 project file.

.js A .js file is a JScript file.

JScript JScript is the Microsoft version of the ECMAScript language. JScript can be used with ORCAstra to automate tests and to define functions in Visual Windows. See also “ORCA” on page 2674 and “Visual Window” on page 2699.

JTAG JTAG is an acronym for Joint Test Access Group, which is a set of specifications that enables board- and chip-level functional verification of a board during production. JTAG also refers to the IEEE1149.1 standard test access port and boundary-scan architecture (TAP/BSCAN).

JTAG chain A JTAG chain is one or more devices connected in a serial chain, where the output of one device feeds the input to the next. Pins on the JTAG connector are the start and end points of the chain.

JTAG ports JTAG ports are pins on an FPGA device that can capture data and programming instructions. In Reveal, they are the pins through which Reveal Logic Analyzer extracts the contents of the trace memory to display in

its graphical user interface. Through the download cable, you can also use the JTAG ports to update the patterns and counters that define a trigger.

junction temperature The junction temperature is the temperature of the die in the device package, in degrees Celsius.

L

latch A latch is a two-state buffer fed by two inputs, D and L. When the L input is low, it acts as a transparent input; in this case, the latch acts as a buffer and outputs the value input by D. When the L input is high, it ignores the D input value.

latch-up Latch-up is a condition in which a circuit draws uncontrolled amounts of current across a pn junction, forcing certain voltages up to a certain level and resulting in a degradation of system performance.

latched input A latched input is an input that captures asynchronous inputs.

lateral capacitance Lateral capacitance is the capacitance between conductors on the same layer.

LATTICE LATTICE refers to a type of download cable.

Lattice parallel port The Lattice parallel port is a port for a Lattice download cable.

Lattice Synthesis Engine Lattice Synthesis Engine (LSE) is the synthesis tool that is integrated with the Lattice Diamond software for supported device families, including MachXO, MachXO2, and Platform Manager. See ".ldc" on page 2665.

LatticeCORE IP module A LatticeCORE IP module is a Lattice Semiconductor large modular design block that implements popular industry-standard functions. It can be reused and easily placed within a programmable logic design. Lattice Semiconductor LatticeCORE IP modules are optimized for use with the Lattice Semiconductor device families.

LatticeEC LatticeEC (EConomy) is a family of Lattice FPGA devices characterized by low cost, LUT-based logic, distributed and embedded memory, PLLs and support for mainstream I/Os, and dedicated DDR memory interface logic without dedicated function blocks.

LatticeECP LatticeECP (EConomyPlus) is a family of Lattice Semiconductor FPGA devices that combines the optimized LatticeEC fabric with dedicated high-performance DSP blocks on-chip tailored for the implementation of common DSP functions. It also features LUT-based logic, distributed and embedded memory, PLLs, support for mainstream I/Os, and dedicated DDR memory interface logic.

LatticeECP2 See "LatticeECP2/M" on page 2664.

LatticeECP2/M LatticeECP2/M (EConomyPlus second generation) is a family of Lattice Semiconductor FPGA devices based on 90-nanometer technology and characterized by advanced DSP blocks, high-speed SERDES with PCS (LatticeECP2M family only), high-speed source synchronous interfaces, LUT-based logic, distributed and embedded memory, phase-locked loops (PLLs), delay-locked loops (DLLs), pre-engineered source synchronous I/O support, enhanced sysDSP blocks, and advanced configuration support, including dual boot capabilities.

LatticeECP2MS LatticeECP2MS devices are the same as LatticeECP2M devices but they also include security encryption of the bitstream. See "LatticeECP2/M" on page 2664.

LatticeECP2S LatticeECP2S devices are the same as LatticeECP2 devices but they also include security encryption of the bitstream. See "LatticeECP2/M" on page 2664.

LatticeECP3 LatticeECP3 is a family of Lattice Semiconductor FPGA devices that offers multi-protocol 3.2G SERDES with XAUI jitter compliance, DDR3 memory interfaces, DSP capabilities, high density on-chip memory and up to 149K LUTS, with low power consumption.

LatticeMico System LatticeMico System is a set of tools that are used to implement a soft microcontroller and attached peripheral components in a Lattice FPGA device. It is based on the Eclipse C/C++ Development Tools (CDT) environment, an industry open-source development and application framework for building software. The two integrated tools in LatticeMico System are used in combination with the Lattice Diamond software to coordinate the building of an embedded processor system on an FPGA device and write the software to drive it. See also "MSB" on page 2671 and "SPE" on page 2689.

LatticeSC LatticeSC is a family of Lattice Semiconductor FPGA devices characterized by a high-performance FPGA fabric, high-speed SERDES with embedded advance PCS (physical coding sub-layer), high-performance I/Os, up to 7.8 megabits of sysMEM embedded block RAM, and dedicated logic to support system level standards, such as RAPIDIO, SPI4.2, SFI-4, UTOPIA, XGMII, and CSIX. The devices in this family feature clock multiply, divide, and phase-shift PLLs, numerous DLLs, and dynamic glitch-free clock MUXs.

LatticeSCM LatticeSCM is a family of Lattice Semiconductor FPGA devices characterized by the same features as LatticeSC devices but that also include MACO (masked array for cost optimization): up to 12 embedded structured ASIC blocks per device with a variety of pre-engineered IP blocks.

LatticeXP LatticeXP (eXpanded Programmability) is a family of Lattice Semiconductor FPGA devices that combines the optimized FPGA architecture used in the LatticeEC (EConomy) family with the ispXP technology in a single low-cost chip. It is implemented on a low-cost, 130-nm flash process to provide instant-on, high-security, non-volatile, and reconfigurable features.

LatticeXP2 LatticeXP2 (eXpanded Programmability second generation) is a family of Lattice Semiconductor FPGA devices that combine a look-up table

(LUT)-based FPGA fabric with flash non-volatile cells in an architecture referred to as flexiFLASH. FlexiFLASH features instant-on, small footprint, on-chip storage with FlashBAK-embedded block memories and Serial TAG memory and design security. LatticeXP2 parts also support live updates with TransFR, 128-bit AES encryption, and dual-boot technologies. They include LUT-based logic, distributed and embedded memory, phase locked loops (PLLs), pre-engineered source synchronous I/O, and enhanced sysDSP blocks.

LBE LBE is an abbreviation for logic block editor, which is a module in EPIC that enables you to edit the logic of a selected component to one of its present configuration modes.

.lci The .lci file is the constraint file, a flat ASCII file that is used to specify optimization options and hardware-specific parameters and is read by the Diamond software.

ldbanno Ldbanno is a command-line program that distributes the physical design information back to the logical design to generate a timing simulation file. The ldbanno program back annotates physical information (for example, net delays) to the logical design and then writes out the back-annotated design in the desired netlist format. The ldbanno program is equivalent to the Design > Back Annotate Assignments command in Lattice Diamond.

.ldc An .ldc file is a constraint file used with the Lattice Synthesis Engine for integrated synthesis. The .ldc file uses the same constraint format as the Synopsys Design Constraint (SDC) language. An .ldc file can be edited in the LDC Editor as well as a text editor. See also “SDC” on page 2686 and “synthesis constraint file” on page 2692.

LDC Editor The LDC Editor is a Lattice Diamond application for editing Lattice Design Constraint (.ldc) files.

.ldf An .ldf file is a Lattice Diamond project file.

level-sensitive signal A level-sensitive signal is an input that only affects a function when it is at 1 or 0, and not when it transitions from one logic value to another. See also “edge-sensitive signal” on page 2651.

LFSR LFSR is an abbreviation for linear feedback shift register, which is a shift register with feedback from two or more points in the register chain. The points are called taps. LSFRs are particularly useful in encryption and decryption applications.

.lib The .lib file, is a collection of library elements compatible with most Lattice FPGA device families. Libraries of symbols can be managed using the Symbol Library Manager in Lattice Diamond.

library A library is a set of macros, such as adders, buffers, and flip-flops, used to create schematic designs.

Linux Linux is the UNIX operating system adapted to personal computers. See “UNIX” on page 2697.

load A load is the resistance, capacitance, or both associated with the inputs of one or more devices driven by an output.

lock constraint A lock constraint locks a component. A pin lock constraint specifies that the current routing cannot be changed or unrouted.

LogiBuilder LogiBuilder is a utility within the Lattice PAC-Designer software that allows you to define a power supply sequence controller and monitor or other control circuits using Power Manager or Platform Manager devices. See also “Power Manager” on page 2679 and “Platform Manager” on page 2678.

logic Logic is one of the three major classes of ICs in most digital electronic systems: microprocessors, memory, and logic. Logic is used for data manipulation and control functions that require higher speed than a microprocessor can provide.

logic analysis Logic analysis is the process of capturing digital data from a device so that you can observe the behavior of the signals and locate any failures in the device. It includes the ability to trigger on a complicated sequence of digital events, copy a large amount of digital data from the device under test through the device’s JTAG ports, and display the data in a graphical user interface.

logic analysis core A logic analysis core is a precharacterized piece of logic (intellectual property) specifically designed for logic analysis. See also “Reveal core” on page 2684.

Logic Block View Logic Block View is a view within the Lattice Diamond software that enables you to examine the logic details of one or selected components that have been placed and routed. Logic Block View can be accessed from Floorplan View, Physical View, and NCD View.

logic cell A logic cell is a predefined group of unconnected components that are replicated across the surface of a gate array. It is also called a basic cell. Sometimes PFUs are referred to as the logic cell because they are the major physical portion of the cell.

logic circuit A logic circuit is a set of components that performs a processing or controlling function.

logic element A logic element is a building block defining the logic in a design. These elements are typically primitives—that is, flip-flops, AND gates, OR gates, NOR gates, and so forth—or macros, which are higher-level combinations of primitives.

logic function A logic function is any Boolean logic expression or set of expressions implemented by using a single logic gate or a complex component containing many logic gates used on an integrated circuit.

logic gate A logic gate is the physical implementation of a logic function.

logic optimization Logic optimization is the process of decreasing the area or increasing the speed of a design.

logic synthesis Logic synthesis is the process of starting from a high level of logic abstraction (typically Verilog or VHDL) and automatically creating a lower level of logic abstraction using a library containing primitives.

logical diagram A logical diagram is a schematic that shows component connections with nets and other components in a functional or logical manner.

lookup table (LUT) A lookup table (LUT) is an architectural element within a LatticeECP/EC PFU for implementing combinational logic or memory.

.lpc The .lpc file is the Lattice Semiconductor parameter configuration file, which contains the configuration parameters that you selected in IPexpress.

.lpf The logical preference file (.lpf) is a post-synthesis source file for storing design constraints known as preferences. This file is automatically generated when you create a new project in Lattice Diamond, and it is used as input for developing and implementing the design. The .lpf file can be edited in one of Diamond's preference-editing views such as Spreadsheet View or in a text editor. See also "Spreadsheet View" on page 2689.

LSE See "Lattice Synthesis Engine" on page 2663.

LSR LSR is an abbreviation for a local set/reset signal. In LatticeECP/EC, LatticeXP, and LatticeSC devices, a low signal on the local set reset (LSR) input pin of the LSR component initializes all flip-flops and latches.

lumped capacitance Lumped capacitance is the total capacitance of a net or segment of a net.

LUT LUT is an acronym for lookup table. See "lookup table (LUT)" on page 2667.

LVDS LVDS is an acronym for low-voltage differential signaling, a high-speed, low-power data transmission standard. LVDS is standardized in the ANSI/TIA/EIA-644-1995 electrical characteristics standard. LVDS is a generic interface standard for high-speed data transmission.

LVPECL LVPECL is an acronym for low-voltage positive emitter-coupled logic. An LVPECL is a digital circuit composed of bipolar transistors in which the emitter ends are wired together. It uses a low-voltage power supply rather than the more common negative power supply.

M

MAC MAC is an acronym for multiply accumulates, which is the FIR operation of multiplying a coefficient by the corresponding delayed data sample and accumulating the result.

MachXO MachXO is a family of Lattice Semiconductor programmable logic devices that combines 256 to 2280 optimized look-up table (LUT) components with Lattice Semiconductor's ispXP technology. It provides high

pin-to-pin performance and instant-on associated with CPLDs and the flexibility of FPGAs. This family provides 73 to 271 I/Os with extensive package options in a low-cost device.

MachXO2 MachXO2 is a family of Lattice Semiconductor programmable logic devices that combines an optimized look-up table (LUT) architecture with 65-nm embedded Flash process technology. MachXO2 devices include hardened versions of commonly used functions such as User Flash Memory (UFM), I2C, SPI and timer/counter. Architectural features are included that help manage static and dynamic power consumption: programmable low swing differential I/Os and the capability of turning off I/O banks, on-chip PLLs, and oscillators dynamically.

MACO MACO is a technology that enables ASIC gates to be put into an FPGA device using a few layers of metal.

MACO block A MACO block is the portion of the MACO hard IP that is assigned to a MACO site. If the MACO IP consumes only one MACO site, the MACO block and MACO hard IP are the same.

MACO default timing model The MACO default timing model is the timing data for the MACO block boundary timing. It includes setup and clock_to_out based on conservative clock injection delay.

MACO hard HDL MACO hard HDL is the HDL that forms a MACO block.

MACO hard IP MACO hard IP is the portion of a MACO IP that is implemented using MACO technology.

MACO logical port A MACO logical port is the port MACO block. It is mapped to a MACO physical port (MACO site pin).

MACO physical port A MACO physical port is the pin on the MACO site.

MACO routing MACO routing is the dedicated routing to and from the MACO site. It is a fast but highly restrictive resource. There are three types: MACO_PIO_routing, MACO_EBR_routing, and MACO_FPGA_routing.

MACO site A MACO site is a physical site on the device where MACO hard IP gates can be placed.

MACO soft HDL MACO soft HDL is the portion of a MACO IP that is implemented using standard FPGA resources, such as PIOs, EBRs, SLICEs, or PLLs/DLLs. It refers to the minimum FPGA design required to interface to the MACO hard IP.

macro A macro is 1) a component composed of nets and primitives, flip-flops, or latches that implements high-level functions, such as adders, subtractors, and dividers. Soft macros and relationally placed macros (RPMs) are types of macros. 2) A control language built into ORCAstra that can be used to automate some operations. See also “ORCAstra” on page 2674.

macro instance A macro instance is a copy of a macro library file inserted in a design file. When you add a macro instance to a design you instantiate

the macro. A design may contain multiple instances of the same library file, and each will receive a unique name. Since the library file is copied into the design file when you instantiate a macro, if you then change the library file the changes will not be reflected in the macro instantiated in the design file. In the EPIC User's Guide, the word macro may be used instead of macro instance. A macro library file will always be referred to as a "macro library file." A macro can only be added if the design file and the macro library file you are adding are of the same device family.

macro library file A macro library file is a file containing the definition of a macro. Macro library files, which have an .nmc extension, are created and edited in EPIC, operating in macro mode. See also "physical macro" on page 2676.

macro mode Macro mode is the set of EPIC options available while you work on a physical macro. To be in macro mode, you must have a physical macro file (.nmc) loaded by either opening or creating one.

makefiles Makefiles are files in LatticeMico System that contain scripts that define what files the make utility must use to compile and link during the build process. There are many makefiles employed in the LatticeMico System build process. The makefile file is the application build makefile, calling all of the other makefiles that allow the generation and build of the platform library and for eventually generating the final executable image.

Manhattan routing Manhattan routing is rectilinear routing, that is, routing in which all the routing tracks are perpendicular to one other.

manual placement Manual placement is the process of placing components and macros in the EPIC Editing area by using the place command. When you use manual placement, you select both a source component or macro and the destination site in which to place the component or macro.

manual routing Manual routing is the process of routing the connections between components in the EPIC Editing area by using the route command. When you use the route command, you specify the path the signal routing will take by selecting the net pins or pinwires to be connected and the routing resources to be used (for example, long lines, local lines). After executing the command, the system routes the path you have specified.

map Map is a command-line program that maps a design to an FPGA. It is equivalent to the Map Design process in Lattice Diamond. The input to the map program is one or more EDIF netlists. The output is a physical design (.ncd file), which is a physical description of the components in the target architecture.

map register retiming Map register retiming is a technique for improving timing by moving registers across logic to shift the path delay from one side of a register to the other side.

mapping Mapping is the process of converting a design represented as a logical network of device-independent components, such as gates and flip-

flops, into a network of device-specific components, such as programmable functional units

master port A master port is a port that can initiate read and write transactions.

master symbol A master symbol is a schematic symbol used for title blocks, logos, revision blocks, and other standardized graphic symbols.

MATLAB/Simulink MATLAB/Simulink is a third-party tool from The Mathworks® that provides an interactive graphical environment and a customizable set of block libraries that enable you to accurately design, simulate, implement, and test control, signal processing, communications, and other time-varying systems.

maximum event counter The maximum event counter in Reveal Inserter and Reveal Logic Analyzer determines the maximum size of the count in the trigger expression. The count is how many times a sequence must occur before a THEN statement.

maximum sequence depth The maximum sequence depth is the maximum number of sequences that can be used in a trigger expression in Reveal Inserter and Reveal Logic Analyzer. Sequences are sequential states connected by THEN operators.

.mcs The .mcs file is a PROM-formatted configuration file in Intel MCS-86 format. It is generated when you run the PROM File process in Lattice Diamond or the promgen command-line program.

MCS-86 MCS-86 is a PROM file format from Intel.

.mem The .mem file is the memory initialization file in ASCII format that contains memory contents for a module. It is the input for FPGA module generation. It consists of lines of addresses with corresponding data, all in hexadecimal format. If a memory file is not used, all the RAM contents are initialized to 0. This file is invisible when you generate FPGA modules with IPexpress.

.men (epic) The .men (epic) file is a customizable EPIC menu bar and pull-down menu definitions file in $FOUNDRY/data, where $FOUNDRY is the path pointed to or defined by the $FOUNDRY environment variable.

Mico System Builder See “MSB” on page 2671.

.mif A .mif file is a memory image file output for FPGA module generation. It is used to initialize memory in SmartModels. This file is invisible when you generate modules with the IPexpress.

mixed mode Mixed mode describes operation in the digital and analog domains, such as mixed-mode simulation.

mixed-mode design Mixed-mode design is a design that consists of both schematic and non-schematic blocks.

mixed-signal Mixed-signal describes electronic circuits that include both analog and digital sections.

ModelSim Lattice FPGA Edition ModelSim Lattice FPGA Edition is a simulation tool from Siemens that supports Verilog and VHDL simulation and verification. It is included with the Diamond software.

module A module is an expanded symbol generated from parameters representing the desired size and operating modes of the symbol.

MPARTRCE MPARTRCE is a multiple place and route (multi-PAR) program that uses a performance score for selecting the best place & route iteration. It is available from the command line.

.mrp The . mrp file is an FPGA report file that contains mapping information. The .mrp file lists any DRC errors found in the design and provides details about how the design was mapped (for example, the schematic constraints specified, the logic that was removed or added, and how signals and symbols in the logical design were mapped into signals and components in the physical design). The .mrp file is generated when you run the Map Design process in Lattice Diamond. See also “map” on page 2669.

.msb An .msb file is an XML-format file output by Mico System Builder (MSB).

MSB MSB is an abbreviation for Mico System Builder, which is an integrated development environment based on Eclipse for choosing components, such as a memory controller and serial interface, to attach to the Lattice Semiconductor 32-bit embedded microprocessor. It also enables you to specify the connectivity between these elements. MSB then enables you to generate a top-level design that includes the microprocessor and the chosen components. It uses the same graphical user interface as C/C++ SPE. See also “SPE” on page 2689.

multiplexer A multiplexer, or mux, is a device that combines several electrical signals into a single signal.

multiplexing/demultiplexing Multiplexing is the process by which two or more signals are transmitted over a single communications channel. Demultiplexing is the separation of two or more channels previously multiplexed.

mux Mux is an abbreviation of “multiplexer.” See “multiplexer” on page 2671.

N

.nam The .nam file is a binary waveform name file.

native circuit description file See “.ncd” on page 2672.

native generic database file See ".ngd" on page 2673.

.ncd An .ncd, or native circuit description, file is a binary-format FPGA post-map physical design database file that is generated by the Map Design process in Lattice Diamond. The .ncd file includes mapping information and, potentially, placement and routing information.

NCD View NCD View is a design view in Lattice Diamond that becomes available after a successful run of Place & Route. NCD View provides a categorized index of synthesized design resources and consumption based on the target device and the in-memory native circuit description (NCD) database.

net A net is 1) A logical connection between two or more symbol instance pins. After routing, the abstract concept of a net is transformed to a physical connection called a wire. 2) An electrical connection between components or nets. It can also be a connection from a single component. It is the same as a wire or a signal.

net check A net check is a physical DRC check in the EPIC software that examines one or more routed or unrouted nets and reports any problems with pin counts, tristate buffer inconsistencies, floating segments, antennae, and partial routes. It can be performed on selected objects or on all of the signals in the design. You can run the net check separately or together with the block check.

net name A net name is the name that identifies a net.

net pin A net pin is a pin that exists on a given net. When a component has been placed but has not yet been routed, the connection between that component's net pins (those pins on the net) and net pins from other components is logical, not physical. The pins are associated with each other, even though there is no electrical connection between them. In the EPIC tool's Editing area, these logical connections making up a net show up as ratsnest lines: direct point-to-point connections between net pins. When the net is routed, electrical connections are made between the net's net pins. In the Editing area, these routed connections appear as lines following the routing resources available on the device (long lines, pinwires, switch boxes, and so forth). See "pin" on page 2676 and "net" on page 2672.

netlist A netlist is an ASCII file describing the logical components in a design and how they are connected. It is basically a list of connectors, a list of instances, and, for each instance, a list of the signals connected to the instance terminals. In addition, the netlist contains attribute information.

Netlist View Netlist View is a design view available in Lattice Diamond that displays the design elements of a post-synthesis native generic database (NGD) netlist. It interacts with Package View for dragging and dropping selected signals to the desired locations on the pin layout to establish LOCATE preferences. See also "Package View" on page 2674.

network A network is a collection of logic elements and the wires (nets or connections) that define how they interconnect.

.ngd An .ngd, or native generic database, file is a binary-format FPGA pre-map logical design database file generated by the Translate Design process in Lattice Diamond. The .ngd file represents the logical design information in the ASCII EDIF netlist.

ngdbuild Ngdbuild is a command-line program that, together with edif2ngd, converts netlists into a generic database that uses native primitive terms that the Lattice Diamond software can implement. It is equivalent to the Translate Design process.

.ngo An .ngo file is a native generic object file, a binary-format FPGA pre-map logical design database generated by the EDIF-to-NGD translation phase of the Translate Design process in Lattice Diamond. It contains all of the data in the input .ngd file, as well as information on the physical design produced by the mapping. The .ngo file typically represents one or more hierarchical branches of a larger logical design.

.nmc An .nmc file is an EPIC physical macro file. In addition to components and nets, the file can contain placement information, routing information, or both. See also “physical macro” on page 2676.

noise Noise is the appearance of unexpected voltage values on a net resulting from transitioning nets or circuitry and coupling through interconnect or substrate parasitics. Noise is usually high-frequency AC.

non-Manhattan routing Non-Manhattan routing is routing that is partially rectilinear but includes tracks that are not vertical or horizontal.

O

.o A .o file is a platform library object file in LatticeMico System that is the compiled output of the library source files. It is also the input for creating the platform library archive files.

octal Octal refers to the base 8 numbering system.

OEM OEM is an abbreviation for original equipment manufacturer, a company that purchases components made by other companies and assembles them into products resold under its own brand name.

one-hot encoding One-hot encoding is a type of state machine design that attempts to increase performance by simplifying decode logic. In one-hot encoding, an individual state register is dedicated to only one state. Only one flip-flop can be active, or hot, at a time. The bit position represents the value. For example, in state machine language, each state is assigned its own storage register (flip-flop), and only one state can be active at a time.

optimization Optimization is 1) The process of decreasing the area or increasing the speed of a design. 2) Part of the design implementation process in Lattice Diamond. The design is re-synthesized with the constraints that you specify.

optimizing placement Optimizing placement is the second stage of the placement process. In this stage, the results of the constructive placement are refined. The first stage of placement is called constructive placement.

ORCA ORCA is an acronym for optimized reconfigurable cell array, a type of FPGA from Lattice Semiconductor.

ORCAstra ORCAstra is a Lattice application that enables you to reconfigure and monitor FPGAs in development boards in real time. ORCAstra gives read and write access to registers that control FPGA configuration, PLLs and DLLs, and registers in logic. With this access you can experiment with different configurations and see the results immediately, without going through any of the design implementation processes.

P

PAC-Designer PAC-Designer is a set of Lattice design entry, simulation, and programming tools that support the Lattice ispPAC families of analog devices, including Power Manager, Platform Manager, and ispClock. See also “Power Manager” on page 2679, “Platform Manager” on page 2678, and “ispClock” on page 2661.

package A package is the ceramic or plastic housing in which an integrated circuit is sealed. Fine gold or aluminum wires connect the chip pads to the larger package pins.

Package View Package View is a view in Lattice Diamond that shows the pin layout of the device and displays the assignments of signals to device pins. Package View interacts with Netlist View for assigning pins, enabling you to drag selected signals to the desired locations on the pin layout to establish LOCATE preferences. See also “Netlist View” on page 2672.

.pad A .pad file is an FPGA report file containing a listing of all programmable interface cells (PIC) used in the design, their associated primary pins, and any assigned preferences. The .pad file is generated when you run the Place & Route process in Lattice Diamond.

pad A pad is a metallized area on an integrated circuit that connects the I/O circuits in the chip to the package or substrate with wires or solder balls. All signals on a chip must enter and leave by way of a pad. Pads are connected to package pins in order for signals to enter or leave an integrated circuit package.

.par A .par file is an FPGA report file generated by the Place & Route process in Lattice Diamond. It summarizes the information from all placement and routing iterations. It also shows the steps taken as the program converges on a placement and routing solution.

par Par is a command-line program that performs FPGA placement and routing. It is equivalent to the Place & Route Design process in the Lattice Diamond software. See also .

parameterized module A parameterized module is a core in which you can set the values of one or more inputs without having to regenerate the module. For example, you can specify the input bit width or whether to create RPMs.

parasitic capacitance Parasitic capacitance is any capacitance causing unwanted effects between conductors or in devices.

partitioning Partitioning is 1) The process of splitting a single design among multiple devices. 2) The maximum level of integration (density concern).

path A path is a connected series of nets and logic elements. A path has a start point and an end point that are different, depending on the type of path. The time taken for a signal to propagate through a path is referred to as the path delay.

path delay Path delay is the time it takes for a signal to propagate through a path.

pattern A pattern is a configuration of data values that defines what an event will be or how the event is recognized by the trigger logic of the Reveal core.

pattern comparator A pattern comparator is the part of the Reveal core that compares the input pattern to the sampled trigger bus data to see if they match. It is connected to a signal that goes from 0 to 1 to indicate that the comparator has made a match.

PCB PCB is an abbreviation for printed circuit board. See “printed circuit board” on page 2680.

.pcf A .pcf file is a Power Calculator project file.

PCM PCM is an abbreviation for programmable clock manager. It is a special function block that is used to modify or condition clock signals for optimum system performance. Some of the functions that can be performed with the PCM are clock-skew reduction, duty-cycle adjustment, clock multiplication, clock-delay reduction, and clock-phase adjustment. If you use PLC logic resources in conjunction with the PCM, many other functions, such as frequency synthesis, are possible.

PCS PCS is an abbreviation for physical coding sublayer, a device layer that is frequently used in defining physical layers for high-speed functions such as the 10 Gb Ethernet and for SERDES applications. The PCS supports advanced system level standards, including RapidIO, Sonet, Gigabit Ethernet, 10 Gb Ethernet, Fibre Channel, and PCI Express.

PCS SERDES PCS SERDES is an abbreviation for physical coding sublayer serializer/deserializer. It is the same as a SERDES. See “SERDES” on page 2687.

period A period is the number of steps in a clock pattern multiplied by the step size.

peripheral capacitance Peripheral capacitance, also called fringing capacitance, is the extra capacitance between the edges of the top plate of a parallel plate capacitor and its bottom plate.

perspective A perspective is a combination of windows, menus, and toolbars in the LatticeMico System graphical user interface that enables you to perform a particular task. For example, the Debug perspective has views that enable you to debug the programs that you created in C++ SPE.

PES PES is an abbreviation for programmer's electronic signature. See "programmer's electronic signature" on page 2681.

PFF PFF is an abbreviation for programmable function unit fast, which is a programmable function unit without RAM.

PFU PFU is an abbreviation for programmable function unit. See "programmable function unit (PFU)" on page 2681.

phase-locked loop A phase-locked loop (PLL) is an electronic circuit with a voltage- or current-driven oscillator that is constantly adjusted to match in phase and therefore lock on the frequency of an input signal. In addition to stabilizing a particular communications channel (keeping it set to a particular frequency), a PLL can be used to generate a signal, modulate or demodulate a signal, reconstitute a signal with less noise, or multiply or divide a frequency. PLLs are more commonly used for digital data transmission but can also be designed for analog information.

physical design file The physical design file is the same as an .ncd file. See ".ncd" on page 2672.

physical macro A physical macro is a logical function that has been created from components of a specific device family. Physical macros are stored in files with the .nmc extension. In addition to components and nets, the file can contain placement information, routing information, or both. A macro can be unplaced, partially placed, or fully placed, and it can also be unrouted, partially routed, or fully routed.

Physical View Physical View is a device layout view in Diamond that provides a more detailed small-component floorplan of your design and enables you to view components such as switch boxes and pin wires.

PIC PIC is an abbreviation for programmable interface cell. See "programmable interface cell (PIC)" on page 2681.

.pin A .pin file is a netlist file containing a generic netlist by pin.

pin A pin can be either a symbol pin or a package pin. A symbol pin, also referred to as an instance pin, is the connection point of an instance to a net. A package pin is a physical connector on an integrated circuit package that carries signals into and out of an integrated circuit.

pin layout file A pin layout file is a customizable report of pin information for a targeted Lattice device. The report is exported from Diamond's Spreadsheet View in delimiter-separated text format. The pin layout file can be as simple as

a list of available pins, pad names, functions, and banks; or it can include details such as differential polarity, type, user assignments, default attributes, and custom column information. You select the types of information you want included in the file, specify the order of presentation, and select the delimiter to be used as the value separator. The pin layout file can be viewed and edited in an external spreadsheet application; it can then be imported back in to Spreadsheet View.

pin migration Pin migration is the process of moving a design's pin assignments from the current device to a device that is compatible, usually a device a of the same package and family. The process involves finding out which pins in your current device will be unavailable or have different functions when you switch devices. Lattice Diamond makes this information available through the "Incompatible Pins" dialog box in Spreadsheet View and Package View.

pinout file A pinout file is a comprehensive report of all device pins in a Lattice targeted device, including available and unbonded pins. It is exported from Diamond's Spreadsheet View as a comma-separated value file (.csv) and can be examined in an external spreadsheet application.

pin-to-pin delay Pin-to-pin delay is delay that is contained within a logic function (intrinsic); that is, it is measured from the transition at the logic gate's input to its corresponding transition at that gate's output. Pin-to-pin delay often is mistakenly used interchangably with point-to-point delay.

PIO PIO is an abbreviation for programmable I/O cell. See "programmable I/O cell" on page 2681.

pipelining Pipelining is the insertion of additional pipeline registers to achieve better clock frequency. Clock latency is introduced in pipelining.

pitch Pitch is the center-to-center distance between two parallel wires on a layer at the closest spacing allowed by the design rules.

Place & Route Place & Route (PAR) is the post-map process in Lattice Diamond that assigns device-specific components to specific locations on the device floorplan and then establishes physical connections to join the components in an electrical network.

placer A placer is a software program that maps logic from a design into specific locations in the target FPGA chip.

placing Placing is the process of assigning the device-specific components produced by the mapping process to specific locations on the device.

platform See "hardware platform" on page 2657.

platform description file See ".msb" on page 2671.

platform library A platform library is a set of files in LatticeMico™ System that contain subroutine code referencing the necessary application files for linking during the build process.

platform library archive (.a) file See ".a" on page 2636.

platform library build The platform library build is an integral part of the managed build process in LatticeMico™ System. Another is the application build. The platform library files contain code that is necessary to the linking during the build process. The platform library build also outputs a platform library archive (<*platform*>.a) file that is referenced by the application build. It allows you to override any default software implementation.

platform library object file See ".o" on page 2673.

Platform Manager Platform Manager is a family of Lattice Semiconductor devices that features programmable analog, with CPLD and FPGA blocks all on one chip to integrate power and digital board management functions. Power management functions include MOSFET OR'ing, hot-swap control, power feed, supply sequencing, monitoring, trimming and margining functions. Digital board management functions include reset distribution, power-on configuration, I2C/SPI interface, fault logging, and glue logic. The Lattice PAC-Designer software is provided as the primary design entry tool for Platform Manager devices. See also "PAC-Designer" on page 2674.

platform settings file The platform settings file is the user.pref file that is generated during the build process in LatticeMico System. It contains information on the platform used by the current project.

PLC PLC is an abbreviation for programmable logic cell. See "programmable logic cell" on page 2681.

PLD PLD is an abbreviation for programmable logic device. See "programmable logic device" on page 2681.

PLL PLL is an abbreviation for phased-locked loop. See "phase-locked loop" on page 2676.

PMI PMI is an abbreviation for parameterized module interface, which is a method of generating the core description of a module or IP once in the HDL code, then changing parameter values in the HDL code when values need to be changed. It is an alternative to regenerating the module or IP whenever values need to be changed.

point-to-point delay Point-to-point delay is delay that occurs between driver gate and load gate (extrinsic); that is, it is measured from the output of the driver gate to the input of the load gate.

polarity Polarity is the orientation of a signal, either positive or negative.

POR POR is an abbreviation for power-on reset. See "power-on reset" on page 2679

POS POS is an abbreviation for product-of-sums. See "product-of-sums" on page 2681.

post-synthesis simulation Post-synthesis simulation is a type of simulation that is usually performed after the HDL code has been expanded

into gates. Post-synthesis simulation is similar to behavioral simulation, since it checks design behavior. The difference is that in post-synthesis simulation the synthesis tool's results are checked. If the post-synthesis and behavioral simulations match, the HDL synthesis tool has interpreted the HDL code correctly.

Power Calculator Power Calculator is a Lattice Diamond application that estimates the power dissipation for a given design. Power Calculator uses parameters such as voltage, temperature, process variations, air flow, heat sink, resource utilization, activity, and frequency, to calculate the device power consumption. It reports both static and dynamic power consumption.

Power Controller Power Controller is a low-power device architectural feature that ensures smooth transitions into and out of standby mode.

Power Guard Power Guard is a low-power device architectural feature that controls dynamic power consumption by blocking signals at the input buffers. Power Guard's biggest impact is in standby mode when it can be used to switch off clock inputs that are distributed using general routing resources.

Power Manager Power Manager is a Lattice Semiconductor family of devices that integrates common board power management functions into a single chip for a wide variety of microprocessors and DSPs. Power Manager devices can be programmed in-system through the JTAG interface. The power management algorithm can be designed using the PAC-Designer software. See also "PAC-Designer" on page 2674.

power-on reset Power-on reset is a device function that guarantees the state of all registers when the device's power is initially turned on. Power-on reset also refers to the circuitry that implements the reset.

PPR PPR is an abbreviation for programmable pullup resistors. See "programmable pullup resistor" on page 2681.

pre-placed component A pre-placed component is a component that is placed in a macro library file.

pre-routed net A pre-routed net is a net that is completely routed in a macro library file. When you instantiate a pre-routed macro into a design, the design may require a substantially longer time to place and route, depending on the route complexity.

Precision RTL Synthesis Precision® RTL Synthesis is a synthesis tool from Mentor Graphics that can be integrated into the ispLEVER suite. It supports logic synthesis for device designs in VHDL or Verilog.

preferences Preferences are constraints that guide the design toward performance goals. They include groups, locations, I/O types, and timing specifications. Preferences are interpreted by the design mapping and place-and-route phases of the flow to assign design logic to physical resources. They can be created and edited using Lattice Diamond's preference-editing views, or they can be created and edited manually in the logical preference file (.lpf) using a text editor.

.prf The .prf file is the physical preference file, an ASCII output of the Map Design process. Preferences saved in the logical preference file (.lpf) are written to the native circuit description (.ncd) and to the .prf file during the map process. The map process rewrites the physical preference file each time it is executed. The .prf file is used as an input file to placement and routing and TRACE.

PRF PRF is an abbreviation for programmable resistor fuses, which are the cells that select the programmable pullup resistors. The resistors are sometimes referred to as PRFs.

primitive A primitive is a basic gate or logic element, such as an AND, OR, XOR, NAND, or NOR gate; inverter; flip-flop; or latch that is a building block of complex logic functions, such as comparators, multiplexers, TBUFs, and decoders. 2) A logic element that directly corresponds, or maps, to one of these basic elements.

printed circuit board A printed circuit board (PCB) is a flat, insulated board of typically 2 to 10 layers to which chips and other electronic components are attached and connected by copper wiring. The main PCB in a system is called a “system board” or “motherboard,” and the smaller boards that plug into the slots in the main board are called “boards” or “cards.”

.prm A .prm file is an FPGA report file that shows address mapping information for generated PROM files. The .prm file is generated when you run the PROM File process in Lattice Diamond.

probing Probing is the process of examining the states of a device.

process A process is the running of concurrent logic blocks. The difference between hardware and software programming is that hardware programming is for concurrent processes and software programming is for linear processes.

process corners Process corners are the conditions under which an integrated circuit is manufactured, such as the temperature and voltage. Process corners are specified for fast, typical, and slow conditions to reflect the amount of current handled by the circuit, whether maximum, average, or minimum, respectively.

product term A product term is a wired NOR of E2 cells that functions as a large AND gate in a sum-of-products realization.

product term disable A product term disable is bits that render a product term permanently low voltage. It is simply a row of cells that are always selected in normal operation. These are different from miser bits that actually disable sense amplifiers to lower power consumption. See also “product term” on page 2680.

product term ground Product term ground (PTG) is the common ground terminal for a row of E2 cells. For most cells, it is not really at ground but is fed into the sense amplifier. On some cell configurations, PTG is also used for programming and verifying. Other cell types have a separate signal called progPTG for programming.

product-of-sums Product-of-sums (POS) is a method of logic representation where the outputs are the product of multiple summing terms. More simply, it is OR-AND logic. By inverting the inputs and outputs, it can be used to represent AND-OR logic for the purposes of reducing logic.

progPTG ProgPTG is the programming product term ground, which is the tunneling source for programming E2 cells.

programmable function unit (PFU) A programmable function unit (PFU) is a block within the LatticeECP/EC device that implements combinational logic, memory, and registers. The core of the architecture consists of PFU blocks that can be programmed to perform logic, arithmetic, distributed RAM, and distributed ROM functions.

programmable I/O cell A programmable I/O cell (PIO) is an architectural element within an FPGA that handles actual input and output functions.

programmable interface cell (PIC) A programmable interface cell (PIC) is a cell within the Lattice Semiconductor FPGA device architecture that contains a group of two PIOs. The edges of the architecture consist of PIC blocks that contain two PIOs connected to sysIO buffers. PICs provide high-speed I/O registers and buffering to support a variety of signal interface standards.

programmable logic cell A programmable logic cell (PLC) is a cell containing four slices to which logic components can be assigned. PLCs are distributed uniformly across an FPGA.

programmable logic device A programmable logic device (PLD) is a prefabricated programmable device containing different types of logic circuits that can be connected in accordance with your specifications. CPLDs and FPGAs are classes of PLDs.

programmable pullup resistor A programmable pullup resistor (PPR) is an internal pullup resistor on device input and output pins that can be turned on or off by programming E2 cells known as programmable resistor fuses, or PRFs. The two terms are frequently used interchangeably.

programmable read-only memory Programmable read-only memory (PROM) is a permanent memory device that is programmed by the customer rather than by the device manufacturer. It differs from a ROM, which is programmed at the time of manufacture. PROMs have been mostly superseded by EEPROMs, which can be reprogrammed.

Programmer Programmer is a tool that supports serial, concurrent (turbo), and microprocessor programming of Lattice devices in PC and Linux environments. Device chains can be scanned automatically using the Programmer graphical user interface. Programmer is integrated into the Diamond software environment, and is also available in a standalone version.

programmer's electronic signature A programmer's electronic signature (PES) is an internal storage area that provides programming parameters to the programming software. On HD devices, this is only necessary for pLSI (non-ISP) devices.

project A project is a collection of all files that are necessary to create and implement a design and download it to the selected device. Each project in Lattice Diamond has a directory, device family, device, and a set of implementations (versions) and strategies. The strategy associated with a given implementation contains options that enable the Lattice Diamond software to display and run the processes appropriate for the targeted device and design flow. In LatticeMico System, a project is the software application code written in C/C++ SPE.

project settings file A project settings file is an .xml file in LatticeMico called settings.xml, which contains information about the parent project and its settings, as well as information on the platform referenced by the parent project.

PROM PROM is an acronym for programmable read-only memory. See “programmable read-only memory” on page 2681.

PROM file A PROM file is a file consisting of one or more bitstream files formed into one or more data streams. The file is formatted in one of three industry-standard formats: Intel MCS86 HEX, Tektronics TEKHEX, or Motorola EXORmacs. The PROM file includes headers that specify the length of the bitstreams, as well as all the framing and control information necessary to configure the FPGAs. It can be used to program one or more devices.

promgen Promgen is a command-line program that converts one or more bitstreams into a PROM-formatted file for PROM configuration. It is the equivalent of the PROM File process in Lattice Diamond. The process takes one or more bitstreams files (.bit or .rbt) and creates a PROM image of the bitstream for PROM configuration. When bitstream files are converted to PROM-formatted files (.mcs, .exo, or .tek), the process will also yield a .prm file or PROM image file that includes a memory map to program the PROM.

propagation delay Propagation delay (symbolized by tpd) is the time required for a digital signal to travel from the inputs of a logic gate to the output. If all other factors are held constant, the average propagation delay in a logic-gate IC increases as the complexity of the internal circuitry increases. Propagation delay has a direct effect on the speed at which a digital device, such as a computer, can operate.

PT PT is an abbreviation for product term. See “product term” on page 2680.

PTD PTD is an abbreviation for product term disable. See “product term disable” on page 2680.

PTG PTG is an abbreviation for product term ground. See “product term ground” on page 2680.

pull-down resistor A pull-down resistor is a device or circuit used to reduce the output impedance of a device. It is often a resistor network that holds a device or circuit output at or less than the zero input level of a subsequent digital device in a system.

pull-up resistor A pull-up resistor is a device or method used to keep the output voltage of a device at a high level. It is often a resistor network connected to a positive supply voltage.

PUR PUR is an abbreviation for power-up set/reset (PUR) signal that, for LatticeECP/EC, LatticeXP, and LatticeSC devices, resets the device on power up as a reset and is active high.

Q

Questa Questa is a simulation tool from Mentor Graphics that supports Verilog and VHDL simulation and verification.

R

radix A radix is the base value in a numbering system. For example, in the binary numbering system, the radix is 2.

RAM RAM is an acronym for random-access memory. See “random-access memory” on page 2683.

random-access memory Random-access memory (RAM) is a read/write memory that has an access time independent of the physical location of the data. RAM can be used to change the address values of the function generator that it is a part of.

ratsnest Ratsnest refers to the straight point-to-point lines (flight lines) that indicate connectivity between logic placed on the floorplan.

.rbt An .rbt file is raw bitstream file, which is an FPGA file containing ASCII ones and zeros representing the bits in the bitstream file. If you are using a microprocessor to configure a single FPGA, you can include the RBT file in the source code as a text file to represent the configuration data. The sequence of characters in the RBT file is the same as the bit sequence that will be written into the FPGA. The raw bitstream file differs from the binary bitstream file (.bit) in that it contains design information in the first six lines.

.rcv An .rcv file is an ASCII-based EPIC recovery file.

RC extraction RC extraction is the process of computing an electronic circuit’s resistance and capacitance.

read-only memory Read-only memory (ROM) is a memory chip that permanently stores instructions and data. It retains a state indefinitely, even when the power is turned off. It can be part of a function generator.

rectification Rectification is the process of converting alternating current (AC) to direct current (DC).

reference component A reference component is a component in the macro library file used as a reference when a macro instance is placed, moved, or copied. Placement and routing of all other pre-placed macro components are determined relative to this component. If at least one of the macro's components is pre-placed, the macro will have a reference component. If none of the macro components are pre-placed, the macro will not have a reference component.

region A region is a specified area of the design that is set aside for the placement of physical groups.

register A register is a set of flip-flops in a chip used to store data. It is an accumulator used for all arithmetic operations.

register-transfer level Register-transfer level (RTL) is a high-level hardware description language (HDL) for defining digital circuits in terms of the data flow between registers. It describes circuits as a collection of registers, Boolean equations, control logic, and complex event sequences. VHDL and Verilog are examples of RTL.

Reports View Reports View is a Lattice Diamond application for viewing, but not editing, the various report files generated by the Lattice Diamond software.

resistance Resistance is opposition that a material has to current flow and the resulting dissipation of energy in the form of heat. Resistance is expressed in ohms.

resistor A resistor is an electrical component that limits the flow of current through resistance or impedance. It allows a specific amount of current to flow, as determined by the voltage applied to it.

retiming Retiming is logic optimization to balance the logic (levels) among register pairs to maximize the clock rate.

Reveal core A Reveal core, also called a logic analysis or debugging core, is a precharacterized piece of logic (intellectual property) specifically designed for logic analysis.

Reveal Inserter Reveal Inserter is an application for FPGAs that enables you to select which design signals to use for debug tracing or triggering, then generates a core on the basis of these signals and their usage. After generating the required core, it generates a modified design with the necessary debug connections and links it to the signals.

Reveal Analyzer Reveal Analyzer is a Lattice Diamond application for FPGAs that enables you to capture and examine signals internal and external to the design. It enables you to set trigger configurations and extract Reveal information from a programmed device through the JTAG ports. Its graphical user interface allows you to trace signals and buses in a waveform viewer.

RF RF is an abbreviation for radio frequency, which refers to 1) That portion of the electromagnetic spectrum used for communications 2) The circuitry used to work with high-frequency signals.

rise time Rise time is the time it takes for the leading edge of a pulse to rise from 10 percent of its peak voltage to 90 percent of its peak voltage.

rising edge A rising edge is a signal transition from a logic 0 to a logic 1. It is also called a positive edge.

ROM ROM is an acronym for read-only memory. See “read-only memory” on page 2683.

route-through A route-through is a net that passes through a logic block.

router A router is a software program that connects all appropriate pins to create the design’s nets.

routing Routing is the process of assigning logical nets to physical wire segments that interconnect logic cells in the device.

RTL RTL is an abbreviation for register-transfer level. See “register-transfer level” on page 2684.

Run Manager Run Manager is a Lattice Diamond application that enables you to run multiple implementations (versions) of your design project and compare the results. You can use Run Manager to run multiple synthesis and place and route passes, monitor the progress, view reports, and identify the best implementation.

running Running is the process of executing a software program.

.rva The .rva file is output by Reveal Analyzer. It contains the project name, the core configuration trace and trigger signals, the trigger and event settings, and the Reveal chain acquisition data. This file is also an input file when you re-open a project that you previously saved.

.rvl An .rvl file is a Reveal Inserter project file, which contains the connections for each core and all the settings of the debugging logic.

.rvs An .rvs file is a Reveal Inserter project file, which contains the connections for each core and all the settings of the debugging logic.

Rx channel An Rx channel is the receive channel in a PCS SERDES.

S

sample A sample is a trace data value captured from the trace bus and written into the trace memory in Reveal.

saturation logic Saturation logic is logic in which the transistors are operating at or beyond their fully conducting state.

_sc A -sc file is a Schematic Editor log file.

scan chain A scan chain, or scan shift register, is a set of scannable flip-flops, or scan cells, linked in a chain for testing purposes.

scan net A scan net is a net whose pins are all scan pins.

.sch An .sch file is a Schematic Editor file.

schematic A schematic is a drawing representing a design in terms of user and library components.

Schematic Editor Schematic Editor is an application that enables you to create and edit schematic files that can represent a complete design or any component of a hierarchical design.

Schmitt trigger A Schmitt trigger is a type of comparator circuit that forces certain input signals to conform to the shape of true digital signals, or waveforms, before passing them to other parts of the logic circuit: signals with slow rise times, slow fall times, or both; noisy signals; or analog signals. The trigger for this circuit to change states is the input voltage level. The Schmitt trigger uses two voltage thresholds, a high threshold during low-to-high transitions, and a low threshold during high-to-low transitions. It changes its output state only when the input voltage level rises above the higher threshold, then uses hysteresis to prevent the output from automatically switching back until the input voltage level goes below the lower voltage threshold. A Schmitt trigger stabilizes a circuit by preventing noise from triggering rapid switching back and forth between the two output states when the inputs are close to the threshold.

.scr An .scr file is an ASCII-based EPIC command script file.

script A script is a series of commands that automatically execute a complex operation, such as the steps in a design flow. See also “JScript” on page 2662 and “VBScript” on page 2698.

.sdf An .sdf file is an industry-standard file format for specifying timing information. It is usually used for simulation. It is required by ModelSim / Questa for timing simulation.

SDC SDC is Synopsys’ widely used design constraints format that describes the design intent and surrounding constraints for synthesis, clocking, timing, power, test, and environmental and operating conditions. The Lattice Synthesis Engine can read and process SDC. See also “.fdc” on page 2653 and “.ldc” on page 2665.

SDF SDF is an abbreviation for standard delay format. See “.sdf” on page 2686.

SDR SDR is an abbreviation for single data rate, a memory interface in an SDRAM that captures data on one edge of the clock.

SDRAM SDRAM is an abbreviation for synchronous dynamic random-access memory, which is a type of dynamic random-access memory (DRAM) that can run at much higher clock speeds than conventional memory.

sequence depth The sequence depth is the number of sequences used in a trigger expression in Reveal Inserter and Reveal Analyzer. Sequences are sequential states connected by THEN operators.

sequential function A sequential function is a logic circuit whose output values depend on both its current input values and on a sequence of previous input values. See also “combinational function” on page 2644.

sequential logic Sequential logic is a logic network that is built around flip-flops. Examples of sequential circuits are counters and state machines. Sequential logic circuits are not necessarily synchronous just because they contain clocked elements.

SERDES SERDES is an abbreviation of serializer/deserializer, which is an integrated circuit transceiver that converts serial data to parallel data, and parallel data to serial data. SERDES devices are used in telecommunications systems to facilitate the transmission of parallel data between two points over serial streams, reducing the number of data paths and therefore the number of connecting pins or wires required.

serial boot PROM A serial boot PROM is a method of configuring a Lattice Semiconductor FPGA at startup.

serial PROM A serial PROM is a PROM that is read one bit at a time. See also “programmable read-only memory” on page 2681.

set/reset Set/reset is an operation made possible by the asynchronous set/reset property. This function is also implemented by the global reset STARTUP primitive.

setup time Setup time is the period before the clocking edge during which the data input to a latch or flip-flop must be held constant in order to guarantee that the latched data is correct.

sheet A sheet is a page of a schematic.

shift register A shift register is a register in which data is loaded in parallel and shifted out of the register again. It refers to a chain of flip-flops connected in cascade.

sidewall capacitance Sidewall capacitance is the same as lateral capacitance. See “lateral capacitance” on page 2663.

signal A signal is a wire or a net. See also “net” on page 2672.

signal integrity Signal integrity is a signal’s freedom from noise and unpredictable delay caused by increasing coupling capacitance between neighboring signals.

signature a digital signature guarantees the authenticity of an electronic document of message in digital communication and uses encryption techniques to provide proof of original and unmodified documentation.

simulation Simulation is the process of verifying the logic and timing of a design.

simultaneously switching outputs (SSOs) Simultaneously switching outputs (SSOs) are multiple output drivers that change state at the same time, which can result in power supply disturbances or noise.

site A site is an unused or vacant resource on a device where a component can possibly be placed.

skew See “clock skew” on page 2644.

skin effect The skin effect is the tendency of alternating current to flow near the surface of a conductor.

slack Slack is the difference between the expected delay and the required delay. A positive slack time in a static timing analysis report indicates that the delay is acceptable, but a negative slack time indicates that the timing of the design does not meet the timing requirements.

slave port A slave port is a port that cannot initiate transactions but can respond to transactions initiated by a master port if it determines that it is the targeted component for the initiated transaction.

slew rate The slew rate is the speed with which the output voltage level transitions from low to high or high to low. The slew rate determines how fast the transistors on the outputs change states.

slice A slice is an architectural element within an FPGA consisting of two LUT4 lookup tables that feed two registers (programmed to be in FF or latch mode) and some associated logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7, and LUT8. It also includes control logic to perform set/reset functions (programmable as synchronous/asynchronous), clock select, chip select, and wider RAM/ROM functions. The registers in the slice can be configured for positive or negative and edge or level clocks. There are four interconnected slices per PFU block. Each slice in a PFU is capable of four modes of operation: logic, ripple, RAM, and ROM. Each slice in the PFF is capable of all modes except RAM.

slope Slope is the same as the slew rate. See “slew rate” on page 2688.

SoC SoC is an abbreviation for system-on-a-chip, which is an integrated circuit that includes all the components of a computer: microprocessors, memories, logic, and I/O interfaces.

soft macro A soft macro is the logical design that determines how logical elements are interconnected functionally, but not the physical or actual wiring pattern on a chip. A soft macro can be altered with a simple text editor. Such macros are provided in synthesizable Verilog or VHDL. But it is virtually impossible for the IP provider to predict the timing of the macro, even if the customer does not modify it. The process of placement and routing is simply too unstable. See also “hard macro” on page 2657.

software application code The software application code is the code that runs on the LatticeMico32 microprocessor or LatticeMico8 microcontroller to control the components, the bus, and the memories. The application is written in a high-level language such as C++.

SOP SOP is an abbreviation for sum-of-products, a method of logic representation in which the outputs are the sum of multiple product terms. More simply, it is AND-OR logic.

Source Editor Source Editor is a Lattice Diamond application that enables you to create and edit text-based files, such as HDL files, test stimulus files, and project documentation files.

SPE SPE is an abbreviation for Software Project Environment, one of the integrated environments of LatticeMico System. The Lattice Software Project Environment enables you to develop the C or C++ application code that runs on platforms created with Mico System Builder (MSB), use the debugger to validate the code, and deploy the application code to on-chip or flash memory. SPE uses a GNU C/C++ tool chain (compiler, assembler, linker, debugger, and other utilities such as objdump) optimized for the LatticeMico process. It uses the same graphical user interface as Mico System Builder (MSB).See also “LatticeMico System” on page 2664 and “MSB” on page 2671.

speed grade The speed grade specifies the frequency of the internal clock of a design. The higher the number is, the faster the clock speed. For example, -5 is faster than -3, and 5 is faster than 3.

.spf An .spf file is a simulation project file, a script file produced by the Diamond Simulation Wizard. The .spf script file enables you to run the simulator for your project from Diamond.

SPI SPI is an acronym for serial peripheral interface, a core that allows high-speed synchronous serial data transfers between microprocessors, microcontrollers, and peripheral devices. It can operate either as a master or as a slave.

SPI serial flash SPI (serial peripheral interface) serial flash is a method of configuring a Lattice Semiconductor FPGA at startup.

SPICE SPICE is an acronym for Simulation Program with Integrated Circuit Emphasis, which is a simulation program originally developed at UC Berkeley as a transistor-level circuit simulator. It is now often used to refer to any transistor-level circuit simulator.

SPLD SPLD is an abbreviation for simple programmable logic device, which is a PLD with fewer than 1000 gates. It is also known as a low-density PLD.

Spreadsheet View Spreadsheet View is the main application in Lattice Diamond for creating and editing design constraints known as preferences. Spreadsheet View includes separate sheets for Port Assignments, Pin Assignments, Clock Resource, Route Priority, Cell Mapping, Global Preferences, Timing Preferences, Group, and Misc Preferences. Preferences are written to the logical preference file (.lpf), and can also be edited in a text editor. See also “.lpf” on page 2667.

Spreadsheet View – TPF Spreadsheet View – TPF is a Lattice Diamond application that enables you to modify timing preferences for the purpose of experimentation and timing analysis. It is a part of Diamond's Timing Analysis View and contains sheets for Global Preferences, Timing Preferences, and Group. When you make timing preference changes, the modifications are written to a timing preference file (.tpf). You can create multiple .tpf files and access them from the Analysis Files folder of Diamond's File List view. See also "Timing Analysis View" on page 2694

SRAM SRAM is an abbreviation for static random-access memory. See "static random-access memory" on page 2691.

SRL SRL is an abbreviation for shift register latch, a serial shift register used to access the programming circuitry in Lattice Semiconductor devices.

SSO See "simultaneously switching outputs (SSOs)" on page 2688.

STA See "static timing analysis" on page 2691

standard cell A standard cell is a completed design for a logic function on an integrated circuit. The function can be as simple as a clock circuit or as complex as a microcoprocessor. Since standard cells are made using blank wafers rather than partially fabricated wafers, they are more efficient than gate arrays but more costly.

standard delay format See ".sdf" on page 2686.

standard encoding Standard encoding is a type of state machine encoding that forms clusters of states and uses binary encoding for each cluster. One-hot encoding is a special case of standard encoding in which each cluster contains exactly one state. Binary encoding is a special case in which all states belong to a single cluster.

STAPL STAPL is an abbreviation for Standard Test and Program Language. A STAPL file is an output data file generated from a JEDEC input file by the Lattice Deployment tool. See also "Deployment Tool" on page 2648.

state A state is the set of values stored in the memory elements of a device that represent the state of that device at a particular point. To each state there corresponds a specific set of logical values.

state diagram A state diagram is a pictorial representation of the outputs and required inputs for each state transition, as well as the sequencing between states. Each circle in a state diagram contains the name of a state. Arrows to and from the circles show the transitions between states and the input conditions that cause state transitions. These conditions are written next to each arrow.

state machine A state machine is a set of combinatorial and sequential logic elements arranged to operate in a predefined sequence in response to specified inputs. The hardware implementation of a state machine design is a set of storage registers (flip-flops) and combinatorial logic, or gates. The storage registers store the current state, and the logic network performs the operations to determine the next state.

static random-access memory Static random-access memory is a device that holds a value as long as power is continually supplied. It loses its contents when the power is turned off. It is sometimes called volatile memory.

static timing analysis Static timing analysis (STA) is the process of verifying circuit timing by totaling the propagation delays along paths between clocked or combinational elements in a circuit. The analysis can determine and report timing data such as the critical path, setup- and hold-time requirements, and the maximum frequency. The primary advantage of timing analysis is that it can be run at any time and requires no input test vectors, which can be very time-consuming and tedious to create. Although timing analysis does not give you a complete timing picture, it is an excellent way to quickly verify the speed of critical paths and identify performance bottlenecks. See also “timing simulation” on page 2694.

static timing analyzer A static timing analyzer is a tool that analyzes the timing of the design on the basis of its paths. See also “TRACE” on page 2695 and “Timing Analysis View” on page 2694.

Steiner tree A Steiner tree is the shortest route for connecting a set of pins.

strategy In Lattice Diamond, a strategy is a collection of settings for controlling the different stages of the design implementation process, including logic synthesis, mapping, and place and route. Each implementation (version) of a design project in Lattice Diamond is associated with a specific strategy. A default strategy, Strategy1, is automatically assigned to a new project. A strategy can also be a predefined strategy or a customized strategy, and any number of them can be applied to a project to find the best solution to meet design objectives. See also “implementation” on page 2659.

.sty An .sty file is a Lattice Diamond strategy file.

.svf An .svf file is a file in serial vector format, which is a software language that is tailored to drive test access port controllers. An .svf file is an ASCII file that stores programming data for programming one or more fixed algorithm devices in automatic test equipment-type programming environments.

SVF SVF is an abbreviation for serial vector format, which is a software language that is tailored to drive test access port controllers. See also “TAP” on page 2693.

SWL SWL is an abbreviation for soft-wired lookup table, which is a fast, programmable connection between the LUTs in an PFU that allows for very wide combinational functions. See also “lookup table (LUT)” on page 2667.

_sy A -sy file is a Symbol Editor log file.

.sym An .sym file is a Symbol Editor file.

symbol A symbol is a graphical representation of one level of hierarchy.

symbol attributes Symbol attributes are characteristics or properties associated with a symbol.

Symbol Editor Symbol Editor is an application that comes with a standard symbol library. Use Symbol Editor to create symbols or primitive elements that represent an independent schematic module. You can also use Symbol Editor to create decorative symbols, such as title blocks.

symbol origin The symbol origin is the point on a symbol that attaches to the cursor when the symbol is placed in Schematic Editor.

symbolic state machine A symbolic state machine is a state machine that makes no reference to the actual values stored in the state register for the different states in the state table. The software determines what these values should be. All that is defined in a symbolic state machine is the relationship among the states in terms of how input signals affect transitions between them, the values of the outputs during each state, and in some cases, the initial state.

synchronous Synchronous describes an entity that is synchronized to a shared signal like a clock.

synchronous clock A synchronous clock is a synchronous control in which flip-flops are set or reset on the rising edge of the clock.

Synplify Pro for Lattice Synplify Pro® for Lattice is a synthesis tool from Synopsys that is integrated into the suite of the Diamond software tools. It supports logic synthesis for device designs in VHDL or Verilog. It can be invoked from the main window or run as a stand-alone process. Synplify starts with high-level designs written in Verilog or VHDL hardware description languages (HDLs) and converts the HDL into small, high-performance design netlists that are optimized for Lattice Semiconductor devices.

synthesis Synthesis is the process of translating a high-level RTL design description (which might consist of state machines, truth tables, Boolean equations, or all three) into a process-specific gate-level implementation.

synthesis constraint file A synthesis constraint file is a source file that describes the design intent and surrounding constraints for synthesis, clocking, timing, power, test, and environmental and operating conditions. See also “SDC” on page 2686, “.fdc” on page 2653, and “.ldc” on page 2665.

synthesis package A synthesis package is a fixed library of cells, with each cell containing the implementation details in terms of primitive logic.

sysCLOCK sysCLOCK is a Lattice PLL and DLL technology for managing clock distribution and skew to improve overall system performance. See also “PLL” on page 2678 and “DLL” on page 2649.

sysCONFIG sysCONFIG is a Lattice Semiconductor interface that provides multiple configuration modes, as well as the dedicated ispJTAG port and boundary scan, so that configuration memory can be loaded automatically at power-up, or any time that you want to update the device.

sysDSP sysDSP is a Lattice DSP block used to develop digital signal processing functions.

sysI/O sysI/O is a Lattice I/O buffer that supports a variety of single-ended and differential signaling standards, as well as the DQS strobe signal that is required for interfacing with the DDR memory.

sysMEM sysMEM is a Lattice embedded block RAM that can operate as single-port or dual-port RAM.

T

TAP TAP is an acronym for test access port, which is a four-wire serial interface port used in boundary scan. See also "boundary scan" on page 2640.

TAP controller A TAP controller is a state machine with 16 possible states that controls operations associated with boundary scan cells. The basic operation is controlled through four pins: Test Clock (TCK), Test Mode Select (TMS), Test Data In (TDI), and Test Data Out (TDO). The TCK and TMS pins direct signals between TAP controller states. The TDI and TDO pins receive the data input and output signals for the scan chain. Optionally, a fifth pin, TRST, can be implemented as an asynchronous reset signal to the TAP controller.

TBUF A TBUF is an abbreviation for a tristate buffer, or 3-state buffer. It is a special logic gate that can output three different states: 0, 1, or Z. It is based on a standard buffer, with a control input called an enable added to the logic configuration.

TCK TCK is an abbreviation for the Test Clock pin, one of the pins that controls operations associated with boundary scan cells.

Tcl Tcl is an abbreviation for Tool Control Language, an interpreted string-command language that can be used for a wide range of functions, including manipulating text, creating variables, building loops, developing functions, and making operating system function calls.

TDI TDI is an abbreviation for the Test Data In pin, one of the pins that controls operations associated with boundary scan cells.

TDM TDM is an abbreviation for time division multiplexing, which is a technology that transmits multiple signals simultaneously over a single transmission path. Each lower-speed signal is merged into one high-speed transmission.

.tdo The .tdo file is a batch file output by ModelSim/Questa.

TDO TDO is an abbreviation for the Test Data Out pin, one of the pins that controls operations associated with boundary scan cells.

TekHex TekHex is a PROM file format from Tektronix.

Template Editor Template Editor is a feature of the Lattice Diamond Source Editor application. The Template Editor provides a library of system templates that you can insert into a source file. It also enables you to add your own customized templates to a User Templates folder.

test bench A test bench is a VHDL test stimulus file that specifies the input waveforms for simulation. You can manually create a VHDL test bench file (*.vhd), use a VHDL test bench template file (*.vht), or export a VHDL test bench file with the Waveform Editor. See also “Waveform Editor” on page 2699.

test vector A test vector is a set of input stimulus values and corresponding expected outputs that can be used with both functional and timing simulators.

thermal impedance model A thermal impedance model is a feature of Power Calculator that helps calculate the thermal impedance for a given device when it is mounted on the board. A thermal impedance model covers scenarios related to board sizes, air flows, and heat sinks.

Theta JA Theta JA is a measurement of the thermal impedance between the silicon die and the ambient air within a JEDEC-defined environment. See also “Effective Theta JA” on page 2651.

threshold A threshold is the crossover point when something occurs or is observed or indicated. The CMOS threshold and TTL threshold are examples.

tick A tick is one clock cycle.

Timing Analysis View Timing Analysis View is a Lattice Diamond graphical user interface for the static timing analysis program, TRACE. Timing Analysis View’s main window allows you to view the path delay tables and TRACE report of your timing preferences after placement and routing. Additionally, Timing Analysis View includes a TPF version of Spreadsheet View, which allows you to experiment with different sets of timing preferences through the use of timing preference files (.tpf). See also “Spreadsheet View – TPF” on page 2690.

timing closure Timing closure is the iterative process of identifying timing problems in a design, correcting them, and rechecking the design until the timing is optimal.

timing preference A timing preference is an attribute assigned to a path that specifies a timing requirement.

timing simulation Timing simulation is a method of verifying that the timing of a digital design meets your requirements. It gives you detailed information about gate delays and worst-case circuit conditions. Because total delay of a complete circuit depends on the number of gates the signal sees and on the way the gates have been placed in the device, timing simulation can only be run after the design has been implemented. See also “static timing analysis” on page 2691.

TMS TMS is an abbreviation for the Test Clock pin, one of the pins that controls operations associated with boundary scan cells.

.tokens Tokens are labels that you can assign to trigger unit values in Reveal Inserter and display on the waveforms in the Waveform View tab of Reveal Analyzer. These labels identify all bus values the same way for clarity, without regard to the radix notation in which the values appear.

token set A token set is multiple tokens that apply to a specific bus in Reveal Analyzer. You can define multiple token sets in Reveal Inserter so that you can use different radixes for different buses in Reveal Analyzer.

top-down design Top-down design is a hierarchical design methodology that starts a design with the highest level of abstraction and gradually designs underlying blocks until the complete design is implemented in the target technology. Top-down design is often technology-independent at the highest levels of design abstraction. This methodology enables you to focus on the functions (and their interaction) rather than on the device that implements them. At the same time, you are free to view or modify an individual module.

top-level file The top-level file is the main file of a Lattice Diamond design. It contains design control information and either design equations or references to include files containing design equations.

.tpf A .tpf file is a timing preference file that is used with Lattice Diamond's Timing Analysis View for the purpose of timing analysis and experimentation. See also "Timing Analysis View" on page 2694.

.tpl A .tpl file is a Text Editor template file.

TRACE TRACE is an acronym for timing reporter and circuit evaluator, an FPGA application that provides static timing analysis based on timing preferences. TRACE performs two major functions: 1) Timing verification, which is the process of verifying that the design meets your timing preferences 2) Reporting, which is the process of enumerating input preference violations and placing them into an accessible file. TRACE can be run on completely placed and routed designs or on designs that are placed and/or routed to any degree of completion. The report issued by TRACE depends on the completeness of the placement and routing of the input design. See also ".twr" on page 2697.

trace control block The trace control block is the part of a Reveal core that controls the writing of the data values into the trace memory around the trigger events.

trace frame A trace frame is the same as a sample in Reveal. See "sample" on page 2685.

trace memory The trace memory is where the data samples captured at the rising edge of the sample clock are stored in the Reveal core.

transceiver A transceiver is an electronic device or circuit that both transmits and receives analog or digital signals.

transistor A transistor is a semiconductor device that allows or prevents current flow between two terminals, as determined by the voltage or current delivered to a third terminal. It can be used as an amplifier or an electronic

switch. The two major types of transistor are the field-effect transistor (FET) and the bipolar junction transistor (BJT).

transition time The transition time is the same as the slew rate. See “slew rate” on page 2688.

translation See “design translation” on page 2648.

trce Trce is a command-line program that generates reports that can be used for static timing analysis. It is equivalent to the Map Trace and Place & Route Trace processes in Lattice Diamond.

trigger A trigger is a specific state or range of states that causes a trace to be stored for review and debugging in Reveal Analyzer.

trigger expression A trigger expression is a combinatorial or sequential equation of trigger units or both used in Reveal Inserter and Reveal Analyzer.

trigger out A trigger out is a trigger output signal in Reveal Inserter and Reveal Analyzer. It can be a signal in a core that can be connected to the trigger input signal of another core or an I/O signal that can trigger outside the chip or both.

trigger unit A trigger unit is the mechanism for defining the trigger signals and conditions that initiate the collection of data on the trace signals for each logic analysis core in Reveal Inserter and Reveal Analyzer. A trigger unit consists of the signals in the trigger bus, a comparator, a value representing the desired pattern of highs and lows on each trigger bus, and a radix for this value.

tristate A tristate is a buffer that places an output signal in a high-impedance state to prevent it from contending with another output signal.

tristate buffer A tristate buffer is the same as TBUF. See “TBUF” on page 2693.

TRST TRST is an abbreviation for the test reset pin, which is an optional pin that can be implemented as an asynchronous reset signal to the TAP controller in boundary scan.

truth table A truth table is a table that defines the behavior for a block of digital logic. Each line of a truth table lists the input signal values and the resulting output value.

TSALLN TSALLN is a global tristate all signal. In LatticeECP/EC, LatticeXP, and LatticeSC devices, it puts all PIC cells on the device in a tristate condition when it is low. When it is enabled in MachXO devices, it can disable all I/Os.

.tw1 A .tw1 file is an FPGA TRACE file that reports timing results for a mapped design. It shows to what extent the timing constraints for a design have been met. Running the Map Design process in Lattice Diamond generates this file. See also “TRACE” on page 2695.

.twr A .twr file is an FPGA TRACE file that reports the timing results for a placed and routed design. The .twr file enables you to determine to what extent the timing constraints for a design have been met. Running the Place & Route Trace process in Lattice Diamond generates this file. See also "TRACE" on page 2695.

Tx channel A Tx channel is the transmit channel in a PCS SERDES.

U

UART A UART is an acronym for universal asynchronous receiver/transmitter, which is a computer component that handles asynchronous serial communication. Every computer contains a UART to manage the serial ports. Some internal modems have their own UART.

.udo The .udo file is a batch file output by ModelSim/Questa.

UES UES is an abbreviation for user's electronic signature, an internal storage area for the end user to store dates, lots, revisions, and so forth. It does not affect the logical operation of the device.

UGROUP A UGROUP (universal group) is logical constraint that specifies a group of logical components that are to be packed close together. A UGROUP instructs the mapper not to pack unrelated logic into the same block. The mapping process translates the UGROUP to a placement group (PGROUP) in the physical preference file (.prf). A UGROUP can include PFU, PFF, EBR, and DSP blocks.

unconstrained path An unconstrained path is a path that does not have a timing preference or requirement ascribed to it.

unit delay simulation Unit delay simulation is a simplified form of timing simulation in which every digital gate is assumed to introduce one unit of delay to a signal.

UNIX UNIX is an acronym for Universal Interactive Executive, which is a multi-user, multi-task operating system for workstations.

USB USB is an abbreviation for universal serial bus, a type of download cable.

USF USF is an abbreviation for user security fuse, an E2 cell that can be programmed to protect the internal arrays from being read out of the device. It protects custom patterns.

V

.v A .v file is Verilog test fixture file used by ModelSim/Questa for timing simulation. A .v file is also a Verilog file output by Mico System Builder (MSB) in LatticeMico System.

Vantis Vantis is a type of download cable.

.vbs A .vbs file is a VBScript file.

VBScript VBScript is a subset of Visual Basic. VBScript can be used with ORCAstra to automate tests and to define functions in Visual Windows. See also “ORCAstra” on page 2674 and “Visual Window” on page 2699.

.vcd A .vcd file is a value change dump file whose format is specified in the IEEE 1364 standard. It is an ASCII file containing header information, variable definitions, and variable value changes. You can export .vcd file information from the Reveal Logic Analyzer software. The .vcd file preserves waveform information that can be displayed in third-party tools such as ModelSim / Questa.

vector A vector is 1) The logical state of a set of nodes within a circuit as a function of time. 2) A group of signals that has been renamed for convenience during simulation. It is similar to a bus. “Bus” refers to a group of signals on the schematic, and “vector” refers to a group of signals during simulation.

verification See “timing simulation” on page 2694 and “static timing analysis” on page 2691.

Verilog Verilog is a hardware description language (HDL) used to design and document electronic systems. Verilog HDL allows you to design at various levels of abstraction. All Lattice Semiconductor devices support Verilog HDL design.

.vhd A .vhd file is a VHDL test bench file required by ModelSim / Questa for timing simulation.

VHDL VHDL is an abbreviation for VHSIC Hardware Description Language, which is an industry-standard hardware description language for describing the structure and function of integrated circuits. All Lattice Semiconductor devices support VHDL design.

VHDL library A VHDL library is a library of design sources required by VHDL. VHDL also allows named libraries that can contain one or more files. VHDL design units can access other design units in the same and different libraries by declaring the name of the library and design unit to make visible.

.vho A .vho file is a VHDL netlist file required by ModelSim / Questa for timing simulation.

.vht A .vht file is a VHDL test bench template file. You must edit and rename this file with the .vhd extension before you can use it for simulation. See also “.vhd” on page 2698.

via A via is a connection point between two adjacent metal routing layers defined by a pad in each layer.

.vis A .vis file defines a Visual Window for ORCAstra.

Visual Window A Visual Window is a window in ORCAstra that provides a variety of control and display functions for testing designs in FPGAs. ORCAstra comes with several Visual Windows (main and secondary Visual Windows). You can also create your own, custom Visual Windows. See also "ORCAstra" on page 2674.

.vo A .vo file is a Verilog netlist file required by ModelSim / Questa for timing simulation.

W

watchpoint A watchpoint is a type of breakpoint that stops the execution of a software program whenever the value of a specific expression changes, without indicating where this may occur. A watchpoint halts program execution, even if the new value being written is the same as the old value of the field.

.wav A .wave file is a file containing Waveform Viewer waveforms and trigger information.

waveform A waveform is a graphical representation of a set of simulation transitions that depicts the digital or electrical values of a node on the schematic.

Waveform Editor Waveform Editor is a Lattice Diamond application that enables you to graphically create a test stimulus file by clicking and dragging with the mouse. You see exactly what each waveform will look like, as well as its timing relationship to all the other waveforms.

.wdl A .wdl file is a Waveform Editor display file.

.wet A .wet file is a Waveform Editor database file.

.wir The .wir file is a WIR netlist format file created by Viewlogic.

wire A wire is the physical implementation of a net or a signal. See also "net" on page 2672.

wired logic Wired logic is a wire connection between two gate outputs, providing a specific logic function.

Wishbone Bus A WISHBONE Bus is a free, open-source hardware computer bus that enables parts of an integrated circuit to communicate with each other. It is especially useful in digital systems that require usage of IP cores. The Wishbone Bus encourages IP reuse by defining a common interface among IP cores.

workspace A workspace is a basic Eclipse-based software feature that contains all of your LatticeMico System projects, files, and folders and stores everything in a “workspace” folder. Basically a workspace represents everything you do in the LatticeMico System software, what is available, how you view it, and what options are available to you through the different perspectives based upon your settings.

_wt A -wt file is a Waveform Editing Viewer log file.

_wv A -wv file is a Waveform Viewer log file.

X

.xcf An .xcf file is a configuration file used by the Diamond Programmer and by ispVM System for programming devices in a JTAG daisy chain. The XCF file contains information about each device, the data files targeted, and the operations to be performed.

.xml An .xml file is a file written in Extensible Markup Language. In LatticeMico System, it contains code declarations called component instance definitions that define the structure of each component. This file resides in the <*install_dir*> components folder. When a build is generated, this information is copied into the platform description (.msb) file by the Mico System Builder (MSB).

XML XML is an abbreviation for Extensible Markup Language, which is a general-purpose markup language used to create special-purpose markup languages for use on the Worldwide Web.

.xnf .xnf is a file in the external netlist format created by Xilinx.

XOR XOR is an exclusive OR gate. On high-density devices, it is a logical XOR gate in the architecture.

Y

yield Yield is the percentage of defect-free (usable) die on a silicon wafer.

Chapter 23

Design Tool Reference

If you want help with a specific tool, start here.

Clarity Designer

Device View

ECO Editor

Floorplan View

IPexpress

Lattice Synthesis Engine

- “Setting Options for Synthesis and Simulation” on page 87
- “LSE Options” on page 1129
- “Coding Tips for Lattice Synthesis Engine (LSE)” on page 173
- “Lattice Synthesis Engine (LSE) Constraints” on page 1339
- “Synthesizing the Design” on page 578
- Lattice Synthesis Engine Tutorial

LDC Editor

main window

- “Managing Project Sources” on page 34
- “Running Processes” on page 70
- “Viewing Logs and Reports” on page 101

Memory Generator

ModelSim

- ModelSim User’s Manual
- ModelSim GUI Reference Manual
- ModelSim Command Reference Manual
- ModelSim Tutorial

 NCD View

 Netlist Analyzer

 Netlist View

 Package View

 Partition Manager

 Platform Designer

 Physical View

 Power Calculator

 Programmer

 Reports

 Reveal Analyzer

 Reveal Inserter

 Run Manager

 Schematic Editor

 Security Setting

 Simulation Wizard

 Source Editor

 Spreadsheet View

 Symbol Editor

 Symbol Library Manager

 Synplify Pro for Lattice

- Synplify and Synplify Pro for Lattice User Guide
- Synplify and Synplify Pro for Lattice Reference Manual
- Synplify and Synplify Pro for Lattice Language Support Reference Manual

Tcl/Tk Documentation

 Timing Analysis View

 Waveform Editor